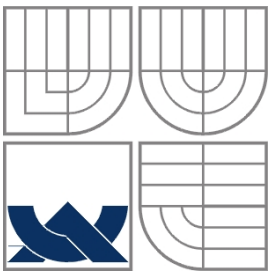
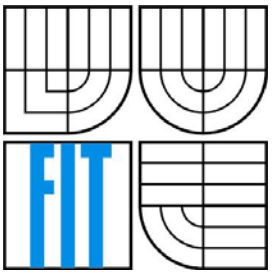


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

IMPLEMENTACE PROGRAMOVÝCH PROSTŘEDKŮ PRO MOBILNÍ ZAŘÍZENÍ

SOFTWARE FOR MOBILE DEVICES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. TOMÁŠ ZOUFALÝ

VEDOUCÍ PRÁCE
SUPERVISOR

ING. IVANA RUDOLFOVÁ

BRNO 2009

Abstrakt

Diplomová práce se zabývá podrobnější analýzou a návrhem implementace systému pro práci s datovými terminály Motorola MC1000. Ty budou použity jako prostředek pro získání informací ze vzdáleného pracoviště. Data jsou z terminálů, ve kterých běží OS Microsoft Windows CE, pomocí dalších částí systému přes SQL databázi a XML rozhraní transformována do ekonomického software, kde probíhá vytváření faktur, zaúčtování pohybů, případně jiné operace. Naopak ten poskytuje vzdáleným zařízením číselníky produktů, skladů a další údaje nezbytné pro provoz. Systém najde uplatnění zejména ve skladech firem, kde bude zboží identifikováno pomocí čárových kódů, které jsou terminály schopné snímat. Pomocí systému bude možno vytvářet výdejky, příjemky, převodky a další doklady. Jejich zapracování do účetnictví je již ponecháno na dalším stupni, tj. ekonomickém software. Celý systém je postaven na míru pro software Pohoda firmy Stormware s.r.o., nicméně nic nebrání jeho využití společně s ekonomickými či jinými systémy dalších výrobců.

Abstract

This master's thesis analyzes and designs implementation of the system for handheld devices like Motorola MC1000. These devices will be used for retrieving information from remote workplace. Data is transformed through SQL database and XML interface by using other parts of the system into economical software, where invoices are created, items are accounted or other operations could be done. This economical software provides some tables like products, warehouses and other information needed for the operation. Best use of the system is in warehouses of companies where products are identified by barcodes, which can be read by handheld device. One can create various documents using this system like conveyances, issue cards, receive cards etc. The next level, which is economical software, is responsible for processing of documents. The whole system is made-to-measure for software Pohoda from Stormware s.r.o. Nevertheless it can be used with another economical, accounting or similar software without any problem.

Klíčová slova

Stormware Pohoda, ekonomický systém, Motorola MC1000, MS SQL, synchronizace, čárové kódy, EAN13, XML, C#, .NET.

Keywords

Stormware Pohoda, accounting system, Motorola MC1000, MS SQL, synchronization, barcodes, EAN13, XML, C#, .NET.

Citace

Zoufalý Tomáš: Implementace programových prostředků pro mobilní zařízení, diplomová práce, Brno, FIT VUT v Brně, 2009

Implementace programových prostředků pro mobilní zařízení

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Ivany Rudolfové.
Další informace mi poskytli pracovníci firmy BHIT CZ s.r.o.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Tomáš Zoufalý
15.5.2009

Poděkování

Na tomto místě bych rád poděkoval lidem z firmy BHIT CZ s.r.o. za pomoc a podporu při řešení projektu. Bez jejich znalostí, zkušeností a trpělivosti by systém nemohl fungovat tak spolehlivě jako v současné době. Zvláště pak děkuji Michalu Kadlčíkovi, který se od samého počátku zabývá testováním a hodnocením odvedené práce.

Obsah

Úvod

Tato diplomová práce pojednává o implementaci systému pro mobilní zařízení, který umožní vytvářet jednoduché účetní doklady. Celý sytém dále spolupracuje s hostitelským systémem, do něhož výsledné doklady importuje a také používá jeho data jako podklad pro tvorbu nových dokladů. Tímto systémem je pro účely práce software Pohoda od společnosti Stormware s.r.o.

Nejdůležitějším přínosem použití datového terminálu MC1000 je možnost identifikovat zboží pomocí čárového kódu, čímž dojde k výraznému zefektivnění a zjednodušení celého procesu tvorby dokladu. Nicméně identifikace pomocí čárových kódů není pro běh aplikace klíčová, a proto tato bude fungovat i na jiných zařízeních, a to zejména kvůli finanční stránce. Terminály Motorola jsou poměrně nákladné zvláště pak v porovnání s dnes běžně dostupnými zařízeními typu PDA, které samozřejmě čtečkou čárového kódu nedisponují.

První kapitola se pokusí uvést čtenáře do kontextu problematiky řešené v této diplomové práci. Je v ní podrobněji popsán celý systém a požadavky na něj.

Druhá kapitola diskutuje možné technologie a popisuje použité.

Ve třetí kapitole jsou zaznamenány nové požadavky a rozdíly v jednotlivých částech systému oproti původnímu návrhu a řeší se zde návaznost na semestrální projekt.

Další kapitola obsahuje detailní popis jednotlivých součástí systému. V prvním úseku se jedná o popis celé komunikační infrastruktury, jejich vlastností a následně pak software ve vlastním mobilním zařízení.

Následující kapitola pojednává o implementaci celého systému, problémech vzniklých při implementaci a jejich řešeních. Jsou zde zmíněny například odlišnosti operačních systému Microsoft nebo řešení ovládání software pomocí dotykové klávesnice s využitím standardních prvků operačního systému.

1 Seznámení s problematikou

Tato kapitola obsahuje jednoduchý popis požadavků na systém a prostředí, ve kterém bude provozován.

1.1 Popis systému

Celý systém je v podstatě dělen na dvě části. Jedná se o komunikační infrastrukturu zajišťující přenos dat od hostitelského systému (Stormware Pohoda) do mobilních zařízení a zpět a potom o vlastní aplikační vybavení mobilního zařízení.

Komunikační infrastruktura sestává s datových úložišť a služeb pracujících nad těmito úložišti. Data mohou být po cestě různě filtrována a modifikována na základě uživatelsky definovaných pravidel a to z několika důvodů. Tak je možné implementovat různá systémová opatření (každá faktura přesahující určitou částku je po zapracování do systému odeslána na email managera, apod.). Dalším důvodem je utajení některých informací. Například situace, kdy z hostitelského systému jsou exportovány nákupní ceny, ale uživatel terminálu (třeba obchodní zástupce) nikdy nesmí vidět nákupní cenu. V poslední řadě je to přizpůsobení jednotlivých terminálů. Každý terminál může mít odlišné požadavky na data, tj. chce jiné balíčky, obsahující typově podobná, ale odlišná data. Například každý obchodní zástupce má své zákazníky a nebo pracuje v určitém regionu.

Softwarové vybavení vlastního mobilního zařízení je aplikace, ve které uživatel vytváří výběrem dat vstupních (rozumí se dat vstupujících z hostitelského systému) a dat vložených na daném zařízení nějaká data výstupní. Výstupními daty jsou většinou účetní doklady. Databázi mobilního zařízení je nutné synchronizovat s komunikační infrastrukturou, čímž se nahrají výstupní data do hostitelského systému (po zapracování infrastrukturou) a stáhnou se nová vstupní data. Mobilní zařízení buď přímo disponuje prostředky pro připojení do IP sítě a nebo je připojeno k PC pomocí Microsoft ActiveSync či novější verze této aplikace, čímž je IP síť simulována. Zařízení má tedy vždy přímou IP konektivitu na infrastrukturu.

1.2 Popis prostředí

Systém bude provozován ve firemním prostředí s požadavky na offline tvorbu účetních či neúčetních dokladů. Offline tvorbou se rozumí tvorba bez přímého napojení na hlavní informační systém firmy. Synchronizace bude tedy probíhat později na požadavek uživatele či automaticky. Uživatel požaduje také tiskové výstupy. Například při tvorbě faktury v terénu potřebuje obchodník zákazníkovi přímo vystavit fakturu či dodací list. Mohou být použity paragonové (většinou přenosné) a nebo klasické laserové či inkoustové kancelářské tiskárny.

Uživatel může, ale také nemusí, pro identifikace výrobků vkládaných do dokladů používat čárové kódy, nejčastěji typu EAN13 [6]. Stejně tak mohou být čárové kódy použity ve skladech pro rychlou identifikaci odběratele. Vyhledávání ve výrobcích musí být ale umožněno i podle jiných kritérií jako je název či kód výrobku.

Terminály slouží pro vytváření dokladů. Modifikace zákazníků či produktů přímo na datovém terminálu není vyžadována zejména z důvodu zhoršeného ovládání tohoto mobilního zařízení.

Komunikační infrastruktura pro synchronizaci dokladů do účetního systému a číselníků z něj musí probíhat bezobslužně. Uživatel nicméně potřebuje možnost celý systém zastavit a pak jej opětovně spustit.

2 Použité technologie a prostředky

Technologie použité v systému je nutné rozdělit na vlastní mobilní zařízení a dále pak na komunikační infrastrukturu, protože jsou vyvíjeny nezávisle a na jiné platformě.

2.1.1 Mobilní zařízení

Datové terminály Motorola MC1000 [3] běží na operačním systému Microsoft Windows CE ve verzi 5.0. Pro vývoj aplikace na této platformě je možné použít různých programovacích jazyků využívajících řízený či neřízený kód. Z oblasti nativních jazyků je to samozřejmě zejména C++ s využitím „Win32“ API na Windows CE. Za druhou kategorii, tedy řízený kód, je zde dostupný Microsoft .NET framework ve verzi Compact. Nejedná se tedy o plnohodnotný .NET Framework, nicméně základní principy, postupy a metody jsou zde stejné. Jazyk pro vytváření takového kódu je dnes možno také volit z poměrně velké škály, nejpopulárnější zástupce je zřejmě C#. V těsném závěsu je pak Visual Basic .NET.

Pro vývoj aplikace na datovém terminálu bude tedy použito jazyka C# a technologie Microsoft .NET Compact Framework verze 2.0. To dává aplikaci také jednu velkou výhodu, kterou je portabilita. Není problém provozovat takto vytvořený software na jiném operačním systému. V tomto případě je jiným systémem myšlen Windows Mobile, který ač je v mnohém podobný Windows CE, tak úplně stejný není. Tento operační systém je jádrem dnes běžně používaných PDA, čímž bude umožněn bezproblémový chod na těchto zařízeních.

Jelikož vybraná platforma nedisponuje žádnou podporou tisku, je nutné řešit tisk jinak. Je samozřejmě možné implementovat protokol dané tiskárny a tiskové výstupy řešit tímto způsobem, ale tento postup je poněkud více časově náročný. Navíc uživatelé mají různé tiskárny a tedy i různé protokoly. Z tohoto důvodu bude lepší využít stávajících ovladačů/aplikací třetích stran. Vhodným kandidátem je PrinterCE.NetCF SDK [12]. Je to komerční knihovna běžící pod .NET frameworkem, která podporuje rozličné tiskárny a protokoly. Tisk pomocí této knihovny probíhá kreslením na canvas, tj. plátno. To spočívá v poskládání obrázku tiskového výstupu pomocí metod pro kreslení čar, grafických útvarů a konečně textu. Tato metoda není nikterak úchvatná, leč na dané technologii rychlá a nejspíše jediná dostupná.

V mobilním zařízení je nutné uchovávat data přijatá z číselníků a také vytvořené účetní doklady. V těchto datech je také potřeba často vyhledávat a řadit je, proto by bylo vhodné použít nějaký SQL stroj. Jelikož se jedná o zařízení s omezenými výpočetními i kapacitními schopnostmi, je vhodné zvolit takzvanou embedded databázi. Jedná se o databázi v jednom souboru, k níž se přistupuje pomocí DLL knihoven a nikoli přes klasický server, tak jako je tomu třeba v případě MS SQL serveru. K disposici je několik databází od různých výrobců. Je to například SQLite for Windows CE, Embedded Firebird a také Microsoft SQL Compact Edition [11]. Poslední jmenovaná

je poměrně značně diskutovaná a nová technologie protlačovaná svým zakladatelem. Z toho důvodu je také očekávána velká podpora i do budoucna. Oproti klasickému SQL serveru od Microsoftu verze CE nepodporuje pohledy a uložené procedury ani triggery. Na oplátku je poměrně malá a sestává pouze z jednoho souboru. Ten lze bez problémů přenášet a otevřít na jakémkoli jiném zařízení a také na klasickém PC. V současné verzi 3.5 SP1 není žádných zásadnějších chyb, proto bude použita pro výše zmíněné účely.

Pro zpracování čárových kódů bude použito knihoven přímo od výrobce zařízení. Ty jsou dostupné pro .NET a obsahují plně obalený engine pro práci s čárovými kódy. V praxi to znamená, že po načtení kódu vrací rovnou informaci v něm zakódovanou ve formě textového řetězce a není nutné nikterak zkoumat kontrolní součty a podobně.

2.1.2 Komunikační infrastruktura

Infrastruktura předávající data mezi hostitelským systémem a mobilním zařízením sestává z několika částí. Všechny mají společné to, že běží na platformě Microsoft Windows. Verze operačního systému může být ovšem různá. Počínaje Windows XP, přes Windows Vista až po serverové edice Windows. Na všech těchto verzích je možné opět využít platformu Microsoft .NET Framework, tentokrát v plné verzi. Vzhledem k tomu, že tato byla využita již pro vývoj v mobilním zařízení a splňuje veškeré potřebné standarty a požadavky, bude ve verzi 2.0 použita pro vývoj v komunikační infrastruktuře. Programovací jazyk bude opět C#. Použitím stejné technologie na obou platformách dosáhneme v některých případech znovupoužitelnosti kódu. Samozřejmě se jedná převážně o méně náročné metody a algoritmy, nicméně jakékoli spojení těchto dvou kategorií vývoje je vhodné.

Data ukládaná při přenosu musí být co nejjednodušeji dostupná. Není nutné složitější filtrování a vyhledávání, nicméně i přes to se jako nejvhodnější varianta úložného prostoru jeví SQL server. Pro takto jednoduchý požadavek jako je prosté datové úložiště zřejmě nebude rozhodující výkon serveru a je v podstatě možné použít jakýkoli z běžně dostupných. Jako optimální řešení se opět naskýtá firma Microsoft se svým SQL serverem. Zákazníkovi se serverovým operačním systémem je možné provést instalaci na plný SQL server, kde může využít pokročilejších metod zálohování. Pokud bude systém provozován bez plného SQL serveru, který je značně finančně náročný, je možné použít i edici SQL Express. Tato edice je zdarma a její omezení nejsou pro daný účel podstatná. Jediný problém může nastat v licenční politice v kombinaci s Windows XP či Vista. Na tyto systémy je možné SQL Express nainstalovat (tak jako plný SQL server), nicméně počet současně připojených zařízení nesmí přesáhnout 10. To je ale pouze softwarově nevynucované omezení vztahující se na zákazníka, který navíc používá více mobilních zařízení. V praxi taková situace běžně nenastane, protože zařízení by se musela připojovat ke komunikační infrastruktuře současně, nicméně na tento licenční problém je třeba zákazníka upozornit.

Do databáze je možné přistupovat přímo z frameworku pomocí dostupných metod a to jak z plné, tak compact verze. Nicméně přístup přímo k SQL serverům celé architektury může být i bezpečnostní problém. Proto pro přístup budou použity webové služby. Ty budou instalovány na Internet Information Services operačního systému. Tím vzniká omezení na OS, ten musí obsahovat IIS. Nelze tedy používat systémy ve verzi Home, které IIS nedisponují.

Předávání dat bude probíhat ve formě XML balíčků. Pro každý typ dat, tj. například adresář, produkty, faktura atd. bude navrženo XSD schéma [2], které přesně definuje strukturu daného balíčku. Podle schématu lze balíček validovat a nemůže dojít k problému se špatně naformátovanými daty. Výhodou je snadný převod dat v striktním formátu daném XSD schématem do databázových tabulek. Další technologií spojenou s XML jsou xpath dotazy a jejich použití při filtrování obsahu balíčků na různých úrovních infrastruktury.

Do systému Pohoda jsou data importována pomocí XML brány tohoto systému. Importované soubory musí být validní podle schémat navržených firmou Stormware, a proto je nutné balíčky ze schémat popsaných výše pomocí XSLT transformací [10] převést.

3 Požadavky a návrh systému

Diplomová práce navazuje na semestrální projekt. V něm byla provedena analýza požadavků a návrh řešení celého systému. Tato kapitola tedy v některých bodech rozšiřuje, popřípadě drobně upravuje, výstupy semestrálního projektu. Nejprve budou nastíněny požadavky uživatele na systém, dále pak návrh architektury celého systému a také vlastní aplikace na mobilním zařízení.

3.1 Uživatelské požadavky

Uživatel od systému očekává možnost snadno a rychle vytvářet požadované doklady bez přímého napojení na hostitelský systém, kterým je Pohoda. Systém musí být v maximální možné míře konfigurovatelný a škálovatelný, do budoucna je očekáváno napojení na jiné hostitelské systémy. Vlastní aplikace musí být modulární, jednotlivá funkcionalita může být rozšiřována nebo naopak omezena dle náročnosti zákazníka (funkcionalitou je myšlena práce nad různými typy dokladů). Aplikace bude pracovat na mobilním zařízení typu PDA, popřípadě zařízení se čtečkami čárových kódů pro urychlení práce. Systém tedy musí s těmito kódy pracovat.

Další podrobnější požadavky budou probrány v následujících kapitolách.

3.1.1 Tiskové výstupy z mobilního zařízení

Uživatel systému chce, aby jako výstup z mobilního zařízení mohl použít tiskové sestavy. Není nutné je nějak modifikovat či doplňovat, stačí pouze statická sestava splňující veškeré zákony. Tento požadavek se týká zejména účetních dokladů jako je například faktura. Zde není možné nikterak měnit tisknutá data, v podstatě je nutné vycházet ze stávajících sestav běžných účetních systémů.

3.1.2 Číselné řady v mobilním zařízení

S předchozím požadavkem úzce souvisí číselné řady implementované přímo v mobilním zařízení. Je totiž nutné zachovat číslo dokladu vytvořené při tisku. Vytištěný doklad může již dostat zákazník a není možné číslo dokladu dále nikterak měnit. Jelikož zařízení pracují na offline principu je nutné, aby každé z nich mělo vlastní číselnou řadu (není možné přidělit stejné číslo dokladu na více zařízeních, ale ty spolu v době činnosti nekomunikují). Číselné řady sestávají z prefixu a/nebo postfixu a vlastního čísla, které je s každým dokladem inkrementováno.

3.1.3 Podporovaná mobilní zařízení

Kromě datových terminálů Motorola MC1000 musí být nově podporovány i vyšší terminály stejného výrobce. To znamená MC3000 (viz. Obrázek 3.1) a MC70 (viz. Obrázek 3.2). Oba tyto terminály

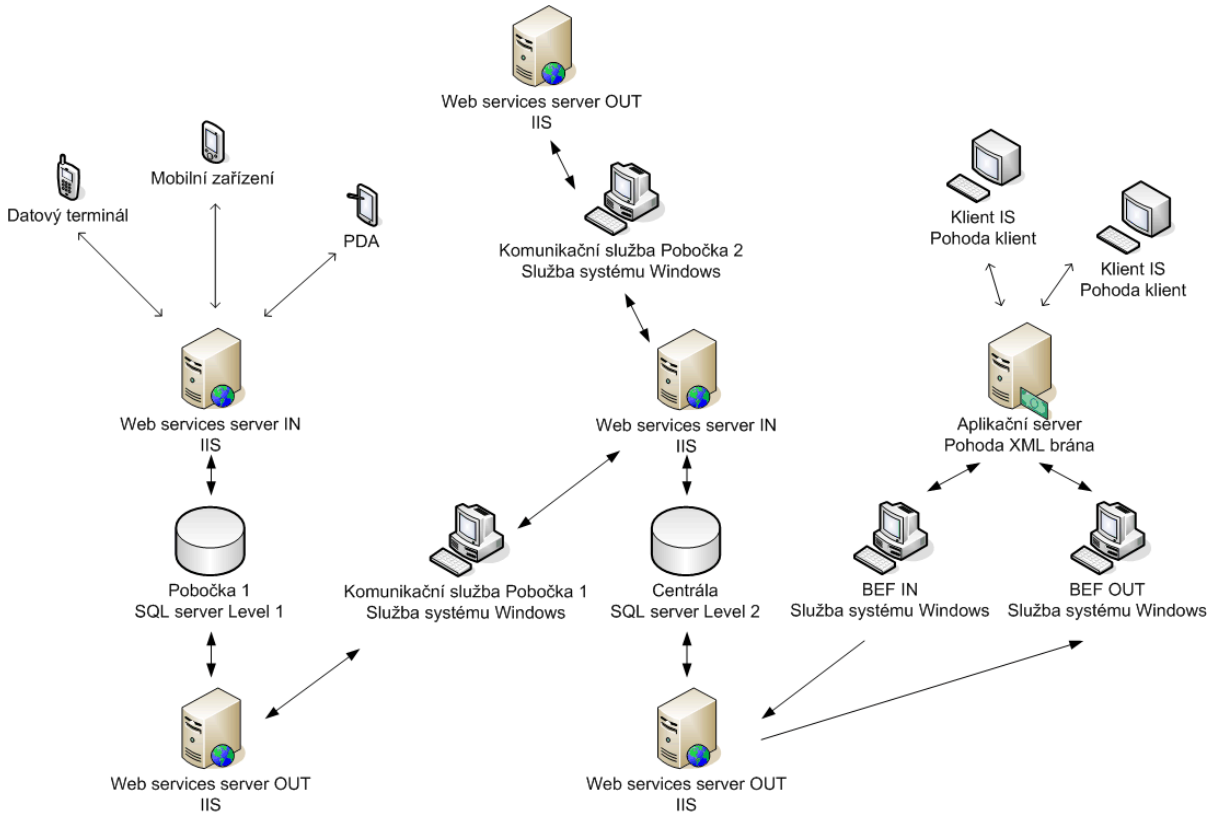
disponují víceméně stejnou technologií, nicméně mají dotykový display. Tím vyvstává nový požadavek na implementaci pokročilejších technik ovládání přizpůsobených právě pro dotykový panel.

Obrázek 3.1– Motorola MC3000

Obrázek 3.2 – Motorola MC70

Dále pak je nutné, aby aplikace fungovala bez problémů na dnešních zařízeních typu PDA. Znamená to odladit aplikaci pro systém Windows Mobile, který má se systémem Windows CE mnoho společného, ale jisté odlišnosti zde jsou. Tyto zařízení většinou neobsahují čtečku čárových kódů a nemají hardwarovou klávesnici. Naopak ale mívají integrovanou podporu pro bezdrátové sítě, bluetooth nebo dokonce GSM modul. Všechny tyto technologie lze využít ke spojení s komunikační infrastrukturou, popřípadě i k tisku na síťové tiskárny bez nutnosti přímého napojení na osobní počítač pomocí technologie ActiveSync.

3.2 Architektura systému



Obrázek 3.3 – schéma komunikační infrastruktury

Komunikační infrastruktura od původního návrhu prodělala několik změn, které byly způsobeny různými podněty.

První změnou v pořadí od mobilního zařízení je způsob synchronizace databáze terminálů s komunikační infrastrukturou. Původní návrh počítal s implementací s použitím frameworku Microsoft Synchronization Services [13], nicméně ten v dané fázi vývoje byl stále ve verzi Beta, a také proto bylo od jeho nasazení upuštěno. Dalším důvodem byla ne úplně dostatečná kontrola nad předávanými daty a slabší dokumentace. Místo této metody byla tedy zvolena jiná alternativa. Jedná se o přístup k datům pomocí prostředníka. Tím je webová služba [2]. Poskytuje pouze jednoduché metody pro uložení a stažení dat a dalších potřebných údajů. Je zde tedy opravdu jen jako přístupová metoda k datovému úložišti a neobsahuje žádnou pokročilou business logiku. Webová služba pro přístup k databázi byla zvolena také kvůli bezpečnosti a škálovatelnosti celého procesu. U IIS totiž není problém na úrovni webového serveru pracovat se zabezpečeným https spojením a také používat jednoduché metody autentizace. Webové služby jsou pak s výhodou použity napříč celým systémem a tedy žádný jiný prvek k SQL databázi neumí samostatně přistupovat.

Komunikační služba mezi centrálou a pobočkou zůstala v podstatě beze změn. Stále slouží primárně k účelu předávání jednotlivých dat bez nějaké logiky, je zde ovšem nově možnost filtrování

a úpravy dat. Proto je také celá architektura navržena dvouvrstvě s pobočkou a centrálou. Na úrovni centrály jsou obsažena všechna data z hostitelského systému, ale na úrovni pobočky už mohou být data upravena pro konkrétní skupinu datových terminálů. Je zde například možné odstranit určitá nechtěná data, která jsou ale zase potřeba u druhé pobočky a nelze je tím pádem filtrovat již na úrovni centrály. V komunikační službě jsou také vytvářeny odlišné balíčky pro různé terminály. Je tedy možné vytvářet nezávislé balíčky ještě i přímo v rámci pobočky.

Z infrastruktury byl vypuštěn předposlední prvek, kterým byl FTP server. Na něm se měly ukládat jednotlivé balíčky před zapracováním do databází hostitelských systémů. Tato metoda vyžaduje další server a použití další technologie. To ale není nezbytné, koncové služby mohou díky webovým službám přistupovat do SQL databáze přímo a na požádání získat potřebná data.

Posledním prvkem jsou Windows služby pojmenované jako Back-end Framework. Jedná se o službu komunikující s datovým úložištěm a hostitelským systémem zároveň. Je rozdělena na dvě části. Vstupní, starající se o data vstupující do systému terminálů (tedy zejména číselníky), a výstupní, jež se stará o data vystupující z celé infrastruktury (vytvořené doklady). Služby jsou odděleny zejména z hlediska robustnosti systému a také z hlediska praktického. Není problém provádět tyto dvě činnosti nezávisle a v jiných časových intervalech. Frameworkem jsou nazvány proto, že jsou navrženy pouze jako rozhraní pro zpracování balíčků a nejsou závislé na konkrétním hostitelském systému. Na tom závisejí až moduly, které framework volá na základě typu zpracovávaných dat. V případě systému Pohoda moduly vždy komunikují s hostitelským systémem pomocí XML brány [9], jež je součástí Pohody.

Celá architektura je tak jako v původním návrhu velice dobře škálovatelná. Vzhledem k tomu, že každá část pracuje samostatně na TCP/IP vrstvě operačního systému, je možné tyto části nasadit nezávisle na sobě na různé počítače stejně tak jako na jediný stroj. Systém sám o sobě nepotřebuje ke své funkci služeb serverového operačního systému a je možné jej provozovat i na stanicích Windows XP a vyšších.

Celý systém pracuje tak, že data typu číselníků jsou z hostitelského systému přes Pohoda Connector a Back-end framework nahrána do datového úložiště na centrále. Při tomto procesu jsou data reprezentována formou XML balíčku s jasně danou strukturou. Po cestě mohou být navíc různě modifikována či promazávána použitím xpath filtrů [8]. Následně se přes komunikační službu dostávají do datového úložiště pobočky. Z mobilního zařízení uživatel po připojení do sítě zvolí synchronizovat, čímž se mu stáhnou tyto XML balíčky. Ty jsou přímo při procesu synchronizace zapracovány do SQL databáze na mobilním zařízení. Zde může uživatel číselníky prohlížet a také je použít pro vytváření různých dokladů. Doklad je po vytvoření uložen do SQL databáze v mobilním zařízení. Následně je pak při synchronizaci převeden na XML balíček (opět s definovanou strukturou) a odeslán do datového úložiště pobočky. Z tohoto místa je pak komunikační službou předán na centrálu. Při procesu předání mohou být data pozměněna opět další sadou xpath filtrů. Back-end framework na centrále po zjištění, že jsou k dispozici nová data daného typu, zavolá odpovídající

modul Pohoda Connectoru, který zapracuje data do hostitelského systému tedy Pohody pomocí XML brány. Odpovědní XML je pak uloženo v archivu společně s původním balíčkem a data se objevují v systému Pohoda.

3.3 Obsluha systému

Komunikační infrastruktura musí fungovat bezobslužně s možností zastavení a obnovení její činnosti. Tato část sestává z webových služeb běžících na IIS a SQL serveru. To jsou pasivní prvky poskytující datové úložiště. Dále pak služby systému Windows pracující s těmito úložišti. Právě tyto služby lze v systému uživatelsky pozastavit či vypnout.

Mobilní zařízení může být v praxi rozlišeno na dvě základní skupiny. Jsou to zařízení s dotykovou obrazovkou a zařízení bez tohoto ovládacího prvku. V jejich obsluze je pak přirozeně podstatný rozdíl.

Motorola MC1000 je zařízení spadající do první skupiny, tedy bez dotykové obrazovky. Má ovšem poměrně dobrou klávesnici s větším počtem kláves. Je to alfanumerická klávesnice, a proto je možno psát jak znaky abecedy, tak čísla a některé symboly. U tohoto zařízení musí být obsluha jednoduchá, intuitivní a navádějící. Bohužel i přes poměrně velkou klávesnici i zde bude nejspíše nutné pro některé akce použít klávesy volané přes funkční klávesu, případně jinou alternativu (například SHIFT či CAPS s příslušnou klávesou).

Naprosto jiná situace je u zařízení s dotykovým panelem. Zde musí jít aplikace ovládat pomocí stylus, tedy pera určujícího polohu myši na dotykové obrazovce. Musí být možné vyvolat grafickou klávesnici a psát znaky včetně diakritiky. Dále pak lehce pomocí klávesnice ovládat celý systém, tedy přecházení mezi jednotlivými obrazovkami atd.

Kombinací předchozích skupin získáváme zařízení typu Motorola MC3000 či MC70. Jedná se o zařízení s dotykovou obrazovkou, nicméně stále obsahují velkou klávesnici.

Ke čtení čárových kódů je využita v zařízeních Motorola integrovaná čtečka. Načítání pomocí kamery na PDA a zpracovávání čárového kódu pomocí OCR zatím nebude podporováno.

3.4 Práce s daty a databází

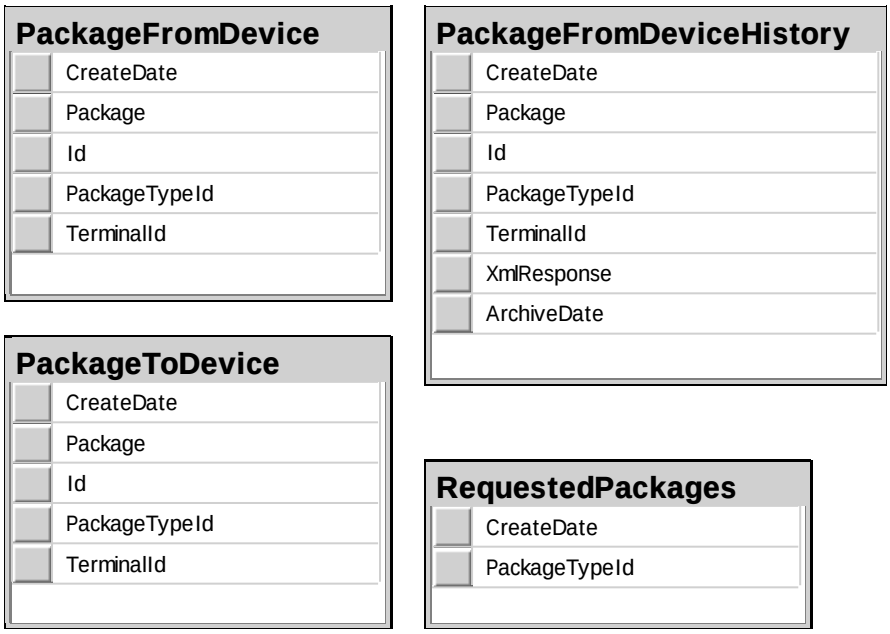
V této oblasti bylo oproti původnímu návrhu učiněno poměrně dost změn. Od striktního návrhu schématu databáze se přešlo k systému dynamicky vytvářené databáze na základě požadavků vlastních dat. Tato změna nastala především proto, aby bylo jednoduše možné pokrýt požadavky různých systémů, systém měl schopnost se sám přizpůsobit novým datům a nebylo třeba s přidáním každého atributu do XML balíčku modifikovat i schéma databáze. Nově bude tedy struktura na mobilním zařízení vytvářena ze striktně definovaných XSD schémat. Znamená to, že pro každý typ dat bude nutné takové schéma definovat. Data jsou pak přes celou infrastrukturu přenášena ve formě

datových balíčků, tedy jednoho XML souboru. Celý tento princip usnadní zejména údržbu celé komunikační infrastruktury. I když by byla databáze navržena obecně, jistě by často s novými požadavky musela být upravována. Tímto návrhem podobné úpravy odpadají, protože s každými daty je po cestě pracováno jako s univerzálním XML balíčkem. Balíčky jsou obecně dvojího druhu.

První kategorie jsou vstupní data z hostitelského systému, například adresář. Ten je přenášen jedním balíčkem obsahujícím veškeré potřebné údaje. Jsou to obecně data číselníkového charakteru. Po cestě v infrastruktuře je na data stále nahlíženo jako na XML soubor a je tedy možné aplikovat XSLT transformace a xpath filtry bez větších obtíží. Mobilní zařízení je první místo, kde je potřeba s daty skutečně pracovat. Zde je nutné převést balíček do odpovídajících tabulek v databázi. To se lehce provede pomocí vytvoření tabulky (pokud již neexistuje) z XSD schématu a nahráním příslušných dat do této tabulky či tabulek.

Druhou skupinou jsou pak data výstupní, tedy jednotlivé doklady. V rámci mobilního zařízení je opět nutné s daty pracovat přímo v databázi, protože doklady je nutné editovat či prohlížet i po vytvoření. V databázi je taktéž dle příslušného schématu vytvořena daná tabulka či tabulky a s daty je v zařízení nad touto databází pracováno. V určitém okamžiku jsou data převedena z SQL databáze na XML balíček. Tímto okamžikem je synchronizace, kdy data opouští terminál a již není potřeba je dále v databázi uchovávat. Po cestě v infrastruktuře se již přenáší pouze jediný XML balíček určitého typu a lze s ním tedy pracovat podobně jako s balíčky vstupních dat.

Databázová struktura datového úložiště sestává ze čtyř jednoduchých tabulek. První je pro data směřující do hostitelského systému (doklady). Další pro opačný směr, tedy data z hostitelského systému (číselníky). Dále pak archivní tabulka pro balíčky obsahující doklady vytvořené na mobilních zařízeních a poslední je pomocná tabulka s typy číselníkových dat, které jsou požadovány z hostitelského systému (viz. Obrázek 3.4).



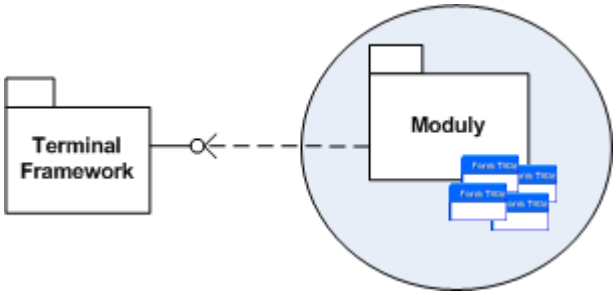
Obrázek 3.4 – schéma databáze v komunikační infrastruktuře

3.5 Návrh struktury aplikace v mobilním zařízení

Aplikace v mobilním zařízení bude dělena do více částí. Hlavní a základní prvek bude Terminal Framework, který poskytuje různé služby dalším součástem. Mezi ně patří hlavně možnost synchronizace databáze a související úkony, grafické prvky a služby grafických formulářů, práce s čárovými kódy či ovládání celého zařízení. Další částí jsou pak moduly, které vytvářejí požadovaná data použitím vlastního uživatelského rozhraní. Databáze bude sestavována na základě aktuální potřeby systému z XSD schématu daného typu dat. Aplikace bude po startu zobrazena ve formě menu, z něhož si uživatel vybere požadovanou akci. Po ukončení se opět vrátí do menu.

Pro potřeby tisku bude implementován mechanismus tiskových sestav, tedy jednotlivé rozdělení na pruhy souvisejících dat jako je hlavička stránky, záhlaví sestavy, položky apod.

Podrobnější popis jednotlivých částí je v dalších kapitolách práce.



Obrázek 3.5 – jednoduchý návrh framework/moduly

4 Popis jednotlivých součástí systému

Následující kapitola obsahuje detailní popis všech součástí systému. Ukazuje způsoby, jakými byly řešeny jednotlivé problémy a návrhy struktur a vlastností systému. Některé části jsou popsány velice podrobně a některé méně zajímavé pouze zevrubně.

4.1 Pomocné třídy

Pomocné třídy neboli helpery obsahují metody, většinou společné pro několik projektů, starající se o určité zařízení či některou oblast. Tyto třídy jsou často sdílené mezi Compact Frameworkem a velkým frameworkem. Některé metody jsou v obou verzích stejné a tedy je možné stejný kód přeložit pro .Net Compact Framework i pro .Net Framework. U jiných tříd je použito funkcí podmíněného překladu v konfliktních úsecích, metody ale stále zůstávají na stejném místě pro obě platformy.

Nejvíce používané a zajímavé třídy jsou zmíněny níže.

4.1.1 LogHelper a EventLogHelper

LogHelper je singletonem obsluhujícím logovací soubor. Z konfiguračního souboru spouštěné aplikace si sám načte umístění log souboru a ve formátu obsahujícím datum vytvoří nový soubor v souborovém systému. Dále pak všem svým uživatelům poskytuje metody pro zápis do tohoto souboru. Pokud to platforma umožňuje (není možné na compact frameworku) zapisuje do logu také další ladící informace jako je posloupnost volaných metod.

LogHelper slouží k ukládání uživatelských informací do logovacího souboru, ale samozřejmě také k ukládání informací o chybách. Poskytuje tedy přímo metody k výpisu chybových hlášení do souboru.

Podobné funkce má i třída EventLogHelper s tím rozdílem, že na systémech Windows pro platformu PC jsou údaje zaznamenány do event logu.

4.1.2 ModulesHelper

Tato třída je základem pro práci s moduly. Jedná se o generickou třídu, která po svém založení automaticky načte seznam všech modulů daného typu z předem definovaného umístění. Moduly jsou DLL knihovny splňující určité podmínky. Ty jsou dány poskytnutou třídou při založení instance objektu, kterou musí všechny moduly dědit a dále lze také v konstruktoru definovat další rozhraní, které musí modul implementovat. Všechny moduly jsou po načtení instancovány a jsou dále dostupné z této třídy.

Moduly mohou pro svou činnost využívat další knihovny, které jsou standardně načítány z podadresáře bin. Struktura, kde jsou umístěny veškeré moduly, je tedy velice jednoduchá a lze se v ní snadno orientovat.

4.1.3 XmlHelper

XmlHelper obsluhuje XML soubory. Stará se zejména o validace souborů či tříd obsahujících načtené XML dle XSD schémat. Validace jako taková je poměrně jednoduchá záležitost, nicméně tato třída umí validovat vstupy i částečně, například pouze jeden strom z poskytnutého XML. Také umí na základě poskytnutého jmenného prostoru ve stromu daný element vyhledat a vrátit. XML soubory umí také spojovat do jednoho.

4.1.4 Ostatní helpery

BindingHelper poskytuje klientům metody pro napárování dat z datového zdroje do uživatelského prvku. Tato třída zná nejběžnější komponenty a umí je nastavit pro mapování dat.

CEMemoryHelper využívá volání nativních funkcí systému Windows CE pro získání informací o využité paměti. Obecně v Compact Frameworku je nutné poměrně často volat nativní funkce, jelikož některé metody nejsou v omezeném rozhraní vůbec implementovány.

SecurityHelper je používán pro práci s MD5 hashí. Metody pro výpočet jsou poskytovány .NET frameworkem.

StringHelper obsahuje metody pro práci s řetězci, které jsou zejména v Compact Frameworku značně omezené. Jsou zde implementovány metody jako Split či Join pracující stejným způsobem jako na velkém frameworku.

4.2 XSD schémata

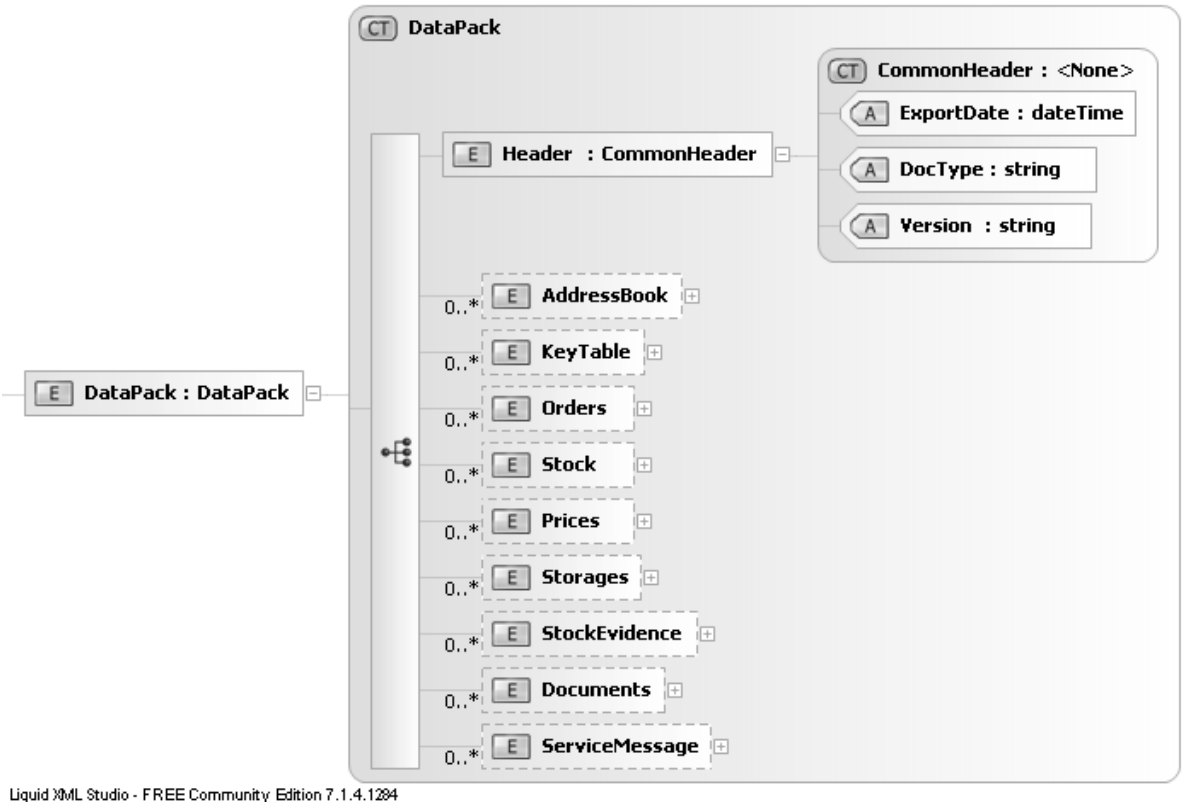
XSD schéma je struktura popisující XML dokument. Obsahuje striktní pravidla jaké elementy dokument může a musí obsahovat a také jakých datových typu (hodnot) mohou tyto elementy nabývat.

Schémata XSD v celém systému lze rozdělit do dvou kategorií. Jedná se o schémata popisující datové balíčky a dále pak schémata pro konfigurační soubory systému.

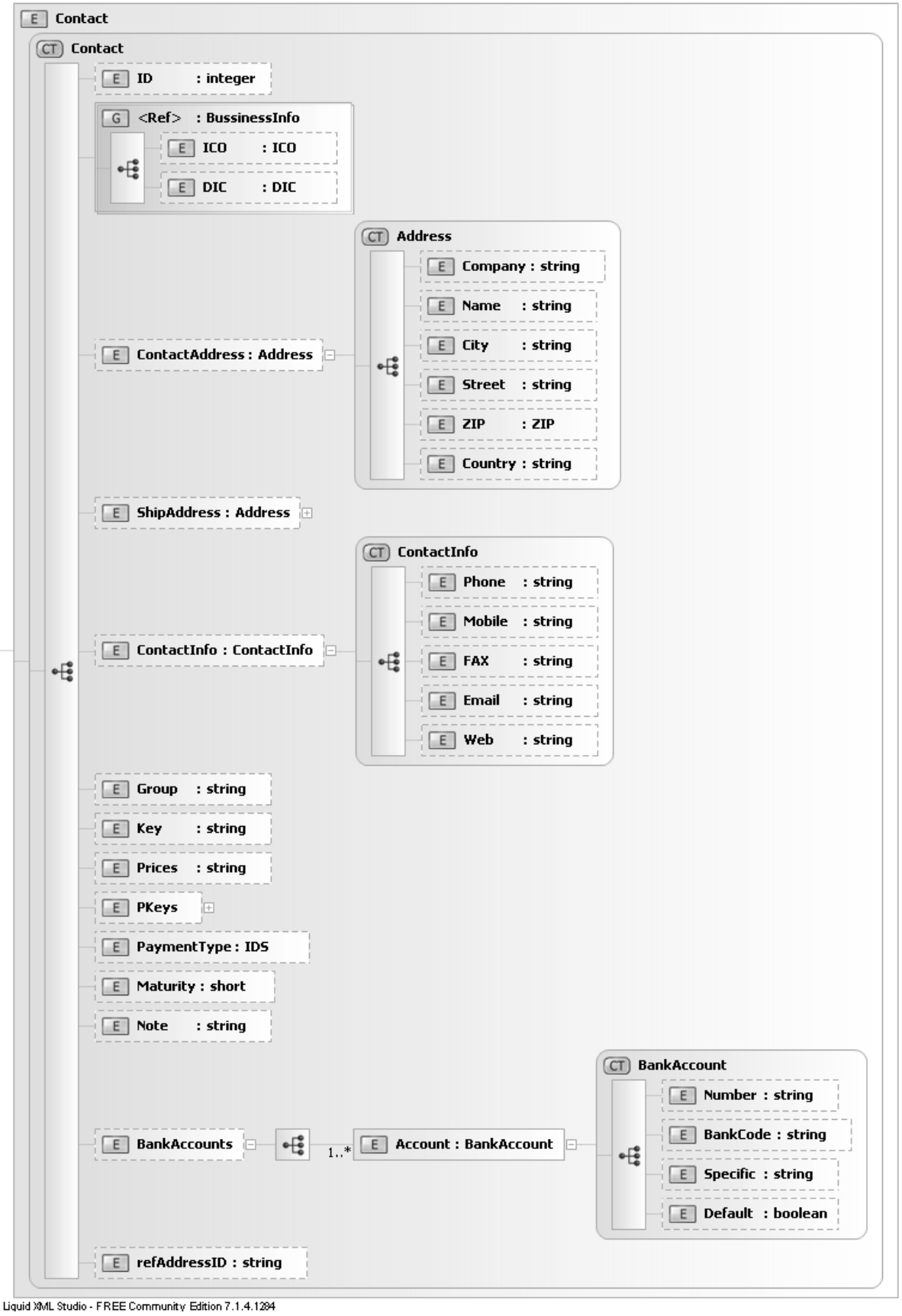
U schémat popisujících konfigurační soubory jednotlivých součástí systémů či modulů jsou použity pokročilejší techniky jako je například omezení unikátnosti určitých elementů ve výsledném XML dle atributů. Všechna schémata jsou soustředěna ve speciálním projektu Schemas. Na základě schémat jsou automaticky generovány příslušné třídy. Ty slouží pro serializaci a deserializaci konfiguračních souborů. V podstatě jde o to, že konfigurační soubory již nejsou zpracovávány jako XML soubory, ale jako objekty. Tyto třídy tedy umí daný objekt načíst a vrátit a také zpětně uložit.

Práce s objekty je potom pochopitelně snadnější i v rámci celého systému a není nutné používat parser pro čtení souborů.

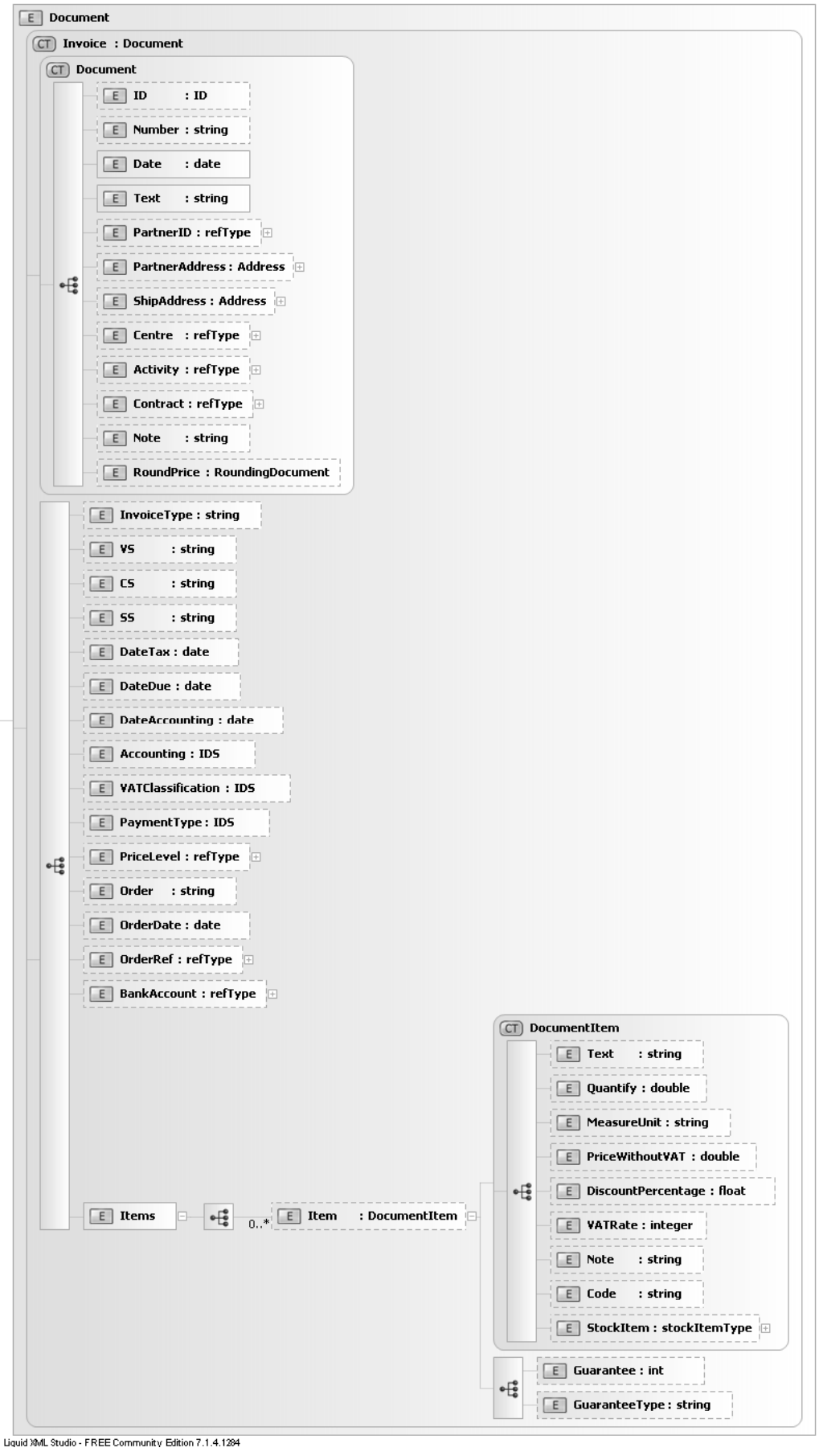
Schémata popisující XML balíčky s vlastními daty systému mohou být dvou typů. Je to buď dokument nebo číselníková vstupní data. Jedná se tedy o popis tabulky číselníku, popřípadě popis dokumentů, většinou s klasickou stavbou hlavička-položky. Tato schémata jsou soustředěna na jednom místě a každý typ dokladu obsahuje také třídu děděnou od společného předka. V rámci této třídy lze dokumenty dle schémat validovat či lze získat dataset se strukturou daného dokumentu. Každý typ má také definován svůj identifikátor pomocí něhož je určován napříč celým systémem. Schémata (resp. konkrétní elementy) jsou od sebe odvozována jednoduchou dědičností. Existuje předek všech dokumentů, který také obsahuje položky. Každý další dokument jej pak doplňuje o své vlastní. Dále pak existuje element DataPack, který zapouzdřuje veškerá další data (viz následující schématické obrázky).



Obrázek 4.1 – XSD schéma datového balíčku



Obrázek 4.2 – XSD schéma kontaktu adresáře



Obrázek 4.3 – XSD schéma faktury

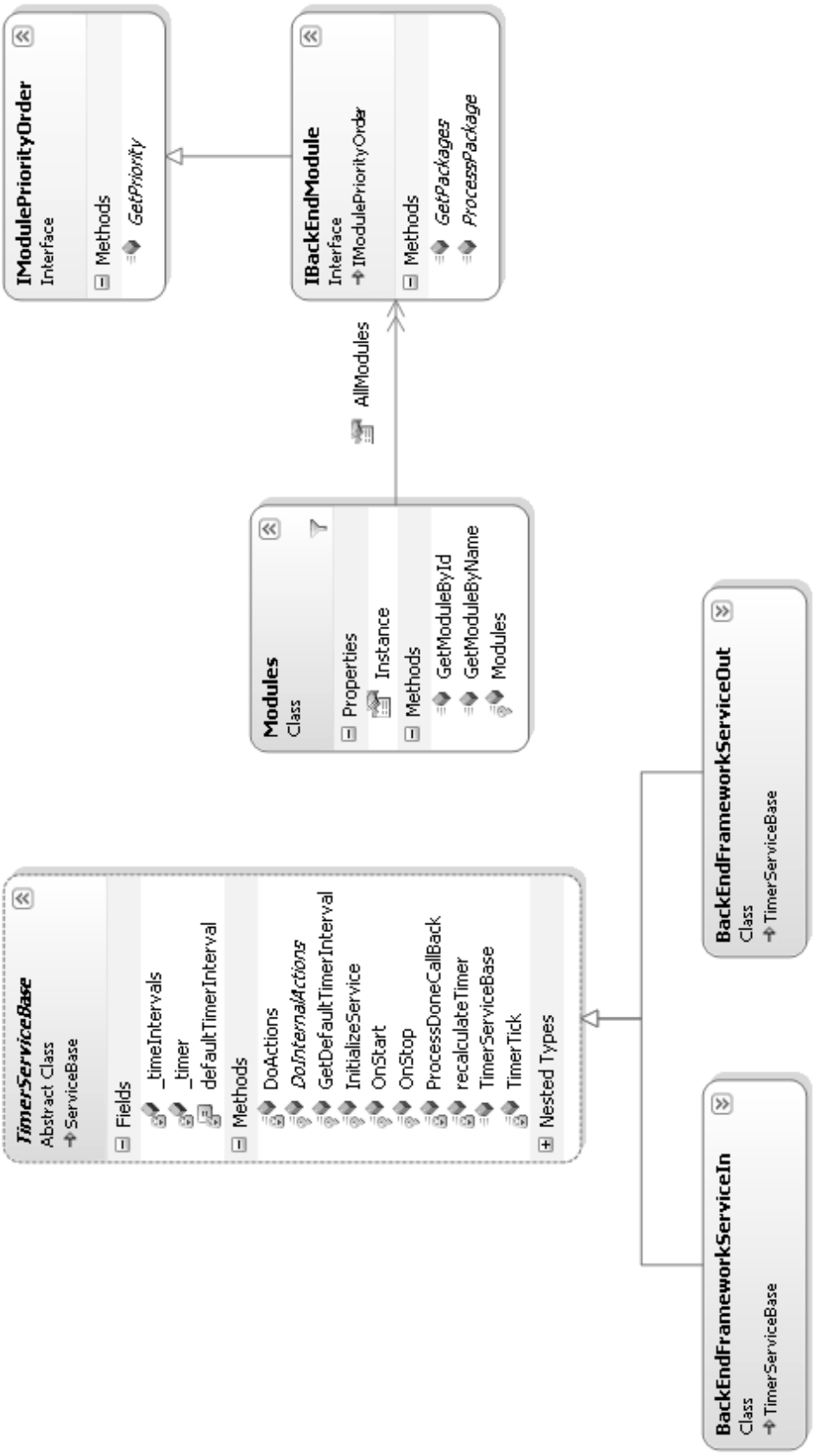
4.3 Back-end Framework

Tato součást architektury slouží pro komunikaci s hostitelským systémem. Existují dvě části a to je část vstupní a druhá část výstupní. První se stará o vstupní data, tedy hlavně číselníky, které jsou stahovány z hostitelského systému. Naopak druhá část do hostitelského systému data nahrává. Princip obou je ale stejný, pro komunikaci s hostitelským systémem využívají knihoven pracujících přímo s konkrétním styčným bodem (v případě Pohody je to XML brána tohoto systému) a na druhém konci komunikují s příslušnou webovou službou.

4.3.1 Vlastní framework

Framework má za úkol načíst při startu potřebné moduly a ty pak v příslušném okamžiku volat. Obě součásti back-end frameworku jsou implementovány jako služby systému Windows, které běží po celou dobu činnosti operačního systému a v určitých intervalech spouští své akce. Intervaly mohou být definovány na úrovni předka systémové služby (viz. Obrázek 4.4). Je zde konfigurováno v jakém časovém úseku má být služba pouštěna a po jaké prodlevě.

Dále framework umožňuje přístup k datům, jelikož přímo do datového úložiště moduly zasahovat nemohou. Jde o to, že poskytuje metodu pomocí níž mohou moduly měnit obsah jednotlivých balíčků. Je zadán typ požadovaného balíčku pro změnu. Všechny takové balíčky jsou pak vráceny z datového úložiště do zpracovávající metody v konkrétním modulu. Výstupem je následně seznam změn, mazaných balíčků a nových dat, což je pak pomocí frameworku opětovně zapracováno do datového úložiště.



Obrázek 4.4 – diagram tříd pro služby Back-end Frameworku a jeho modulů

4.3.2 Moduly

Skutečnou činnost celého back-end frameworku neprovádí sám framework, nýbrž jeho moduly. Pro každý typ dat je implementován modul, který se snaží z hostitelského systému požadované údaje dostat a/nebo je naopak do tohoto systému nahrát. Každý modul musí implementovat rozhraní IBackEndModule. Obecně lze říci, že přístup k modulům v rámci celé architektury je řešen stejně. Vždy existuje singleton, který načítá všechny moduly a pomocí něj se k modulům dá pak přistupovat (viz. Obrázek 4.4).

4.3.3 Pohoda Connector

Pro práci se systémem Pohoda od Stormware s.r.o. slouží součást systému nazvaná Pohoda Connector. Jedná se o soubor metod pro práci s XML bránou zmiňovaného systému. Jako vstup slouží XML balíček ve validním stavu (validním dle XSD schémat). Výstupem jsou pak data, která se objeví v hostitelském systému. XML brána Pohody je realizována spuštěním nové instance systému Pohoda v bezobslužném režimu. To přináší také určitá omezení, se kterými je nutno počítat. Znamená to například, že se vždy spustí hlavní okno programu Pohoda. Také se muže stát (a stává se), že při nevhodném nastavení dojde k zaseknutí celého procesu dialogovou hláškou čekající na potvrzení. Všechny tyto stavy je nutné ošetřit.

Dialogová hlášení jsou ukončována automaticky Pohoda Connectorem. Po 20 vteřinách nečinnosti systému je učiněn pokus o nalezení čekajícího dialogového okna a v případě úspěchu je toto uzavřeno. Dále je pak monitorována aktivita procesu a pokud se zdá po dobu jedné hodiny „zaseknutý“, je ukončen. Bohužel nedokonalost brány Pohody si vyžaduje i tuto drastickou metodu.

Při exportu dat ze systému Pohoda je do XML brány vždy zasílán požadavek na patřičná data. Ta jsou po návratu (XML balíčkem) převedena z formátu Pohody do formátu XSD schémat systému pomocí XSLT transformace. Zde nastává ovšem další problém s nedokonalostí brány Pohody. Velice často se stává, že dává nevalidní výstupy. Je tedy nutné provádět validaci nejprve i podle schémat Stormware, protože v případě nevalidních vstupů nelze očekávat validní výstupy XSLT transformací.

Navíc je do celého procesu opět implementována možnost xpath filtrování a změny v jednotlivých balíčcích a to jak po exportu z Pohody (tedy přímo v balíčku validním dle schématu Stormware), tak po XSLT konverzi. Některé nepotřebné údaje lze tedy odfiltrovat přímo na tomto místě a nemusí se tak zatěžovat kapacity celé komunikační infrastruktury.

Import dat do systému Pohoda je řešen analogicky. Nejprve je provedena validace XML balíčku dle XSD schémat společně s xpath filtry. Dále pak převeden balíček XSLT transformací na strukturu požadovanou Pohodou a zpracován XML bránou.

Back-end framework je připraven zpracovávat i odpovědi z hostitelského systému. Může se totiž stát, že se data z nějakého důvodu nepodařilo naimportovat. Systém Pohoda vrací na každý požadavek do XML brány odpovědní balíček, ve kterém vypisuje výsledky provedené akce.

V současné době není navržen systém zpráv uživateli a tedy není možné při výskytu chyby žádná hlášení podávat a odpovědi jsou pouze ukládány společně s kopií dotazu do archivu. Implementace způsobu vysílání různých varovných hlášení a obecně zpracovávání výstupů z hostitelského systému bude zřejmě předmětem dalšího vývoje.

4.4 Webové služby

Webová služba je aplikace pracující na IIS, tedy na serveru poskytující svůj obsah na protokolu http či https. Tyto služby zajišťují pro systém přístup k datovému úložišti, tedy k SQL serveru. Webové služby jsou pro účely systému implementovány dvě. Jedna slouží pro příjem dat z hostitelského systému a druhá naopak pro odesílání dat do hostitelského systému. Každá z nich má mírně odlišné požadavky a tím pádem jsou zde obsaženy různé metody. V principu ale jde vždy o funkce pro uložení a načtení daného XML balíčku. Navíc je zde implementován mechanizmus, kdy si mobilní zařízení samo určuje, jaké balíčky potřebuje od hostitelského systému. Tedy balíčky jakého typu. Například může být řečeno, že potřebuje jen produkty a adresář, další číselníky ho nezajímají. Je to v podstatě dáno tím, jaké moduly jsou použity v mobilním zařízení a co dané moduly vyžadují pro svou činnost.

Pro posílání balíčků z hostitelského systému byl implementován princip rozdělování posílaných částí. Je v podstatě pracováno s XML soubory, které pro větší databáze hostitelského systému mohou být poměrně velké, a proto je tato funkce žádoucí. Jednou metodou jsou vygenerovány potřebné části balíčku ve webové službě, ta zároveň vrací jakýsi identifikátor této skupiny. Další metodou je možné jednotlivé části stáhnout na mobilní zařízení, kde jsou následně spojeny a je opětovně sestaven rozložený balíček.

Bohužel .NET Compact Framework neumožňuje v základu použití komprimačních metod, tak jako je to možné na plném frameworku. Bylo by tedy možné data bez problému ve webové službě komprimovat, nicméně rozbalení na mobilním zařízení je v použité verzi poněkud problematické. Verze 3.5 již tyto metody umožňuje, a proto s přechodem na novou verzi frameworku bude v mobilním zařízení i webové službě navíc implementován princip komprimace veškerých předávaných informací. Tato metoda má poměrně velkou šanci zásadně snížit velikost přenášených dat. Obecně lze totiž XML soubory komprimovat s poměrně vysokým poměrem komprese.

4.5 Komunikační služba

Komunikační služba slouží k předávání jednotlivých XML balíčků mezi webovými službami centrály a pobočky. V základu tedy nejde o nic složitého. Stačí balíček nahrát z jednoho úložiště a uložit jej do druhého. Tento princip byl ale doplněn možností použití modulů, které mohou podobu balíčku ovlivnit.

4.5.1 Moduly

Podobně jako v jiných součástech architektury jsou i zde implementovány moduly. Při práci s konkrétním typem balíčku je automaticky vyvolán modul (zde jsou moduly vlastně procesory zpracovávající data), který požaduje na svém vstupu daný typ XML balíčku. Takových procesorů může být i více a každý má svou prioritu, která určuje pořadí v jakém jsou jednotlivé moduly na data volány. Existují různé typy modulů a tento princip je také velice jednoduchý pro implementaci nejrůznějších zákaznických modifikací. Například uživatel vyžaduje, aby doklad měl vždy nastaven určitý příznak. V mobilním zařízení nicméně není s tímto příznakem vůbec pracováno a přidání této vlastnosti jen pro konkrétního zákazníka není příliš rozumné řešení. Nicméně na úrovni modulu v komunikační službě není problém toto chování upravit, do každého balíčku typu požadovaného dokladu se prostě příznak přidá (může se jednat třeba o nastavení poznámky v dokladu nebo jiné podobné vlastnosti). Další využití může být při vytváření nových dokladů na základě dokladů stávajících. Z každé faktury vydané, která byla zaplacena v hotovosti, může být na tomto místě vytvořen pokladní příjmový doklad a v hostitelském systému je účetně podchycen i finanční tok z daného dokladu vyplývající.

4.5.2 Filtry

Speciální kategorií modulů v komunikační službě jsou filtry. První požadavek, kvůli němuž původní filtry vznikly, je implementace omezení a ořezání některých balíčků pro danou pobočku. Například v Brně nesmí být vidět sklady Praha. V centrále jsou ale sklady všechny. Proto je na danou pobočku nasazen filtr, který tyto sklady maže. Vlastní filtrování probíhá na základě xpath dotazu, který v balíčku buď nalezená data smaže nebo je naopak ponechá. Pomocí této techniky lze vytvářet velice komplikované kombinované dotazy. Dotazů může být do balíčku učiněno i více.

Dalším případem, kdy jsou filtry použity, je rozlišení obsahu jednotlivých balíčků pro různé skupiny mobilních zařízení v rámci jedné pobočky. Princip je takový, že z jednoho balíčku vznikne více nových balíčků, které jsou přímo určeny na konkrétní zařízení či skupinu zařízení. Každé zařízení může být tedy přesně identifikováno a webová služba pak danému zařízení vrací balíček, jež je pro něj přímo určen.

4.6 Terminal Framework

Software běžící v mobilním zařízení můžeme funkčně rozdělit na několik částí. O hlavní běh celé aplikace se stará Terminal Framework. Ten má za úkol poskytovat základní metody ostatním službám, jež jsou implementovány pomocí modulů. Jednotlivé komponenty frameworku, tak jako i moduly, jsou reprezentovány oddělenými knihovnami a do systému většinou zaváděny na požádání až v okamžiku, kdy jsou potřeba. Celý framework je nezávislý na konkrétním mobilním zařízení,

a proto také nepoužívá žádné metody či SDK knihovny, které nejsou součástí .NET Compact Frameworku nebo nativních volání Windows CE. Závislost na konkrétním zařízení (například čtečka čárových kódů) je řešena použitím modulů.

4.6.1 Vlastní framework

Framework sám kromě zobrazení menu funkčně neumí skoro nic, pouze poskytuje služby dalším částem systému. Je to nicméně nejzajímavější část celého softwarového vybavení na mobilním zařízení a jeho jednotlivé součásti budou tak jak jsou členěny fyzicky (tedy do knihoven) popsány níže.

4.6.1.1 Inicializátor aplikace

Jedná se o úvodní kód načtený při spuštění celé aplikace. Start celého systému probíhá kontrolou, zda-li se nejedná o podruhé spuštěnou instanci. V takovém případě je běh nově spuštěné aplikace ukončen a aktivován je proces předchozí. Pokud kontrola proběhne v pořádku, je zobrazen splash screen, tj. úvodní obrazovka s nápisem čekejte. Mezitím je provedeno zavedení systému a spuštění všech potřebných komponent, kterými jsou Framework, PowerManagement, EventManager a KeyboardManager. Po úspěšné inicializaci je spuštěno hlavní menu aplikace. Jakmile dojde k ukončení tohoto menu, jsou inicializátorem veškeré prostředky obsazené dříve uvolněny.

4.6.1.2 Framework

Framework obsahuje třídu pro synchronizaci dat. Umožňuje synchronizovat vstupní a výstupní balíčky a získání informace o stavu připojení k synchronizačnímu serveru. Dále pak jsou zde metody pro vlastní připojení k webovým službám. Je to jediné místo, které má přes IP vrstvu za úkol komunikovat s infrastrukturou.

4.6.1.3 Framework.PowerManagement

PowerManagement má na starosti správu napájení na mobilním zařízení. Jelikož zařízení může být často nastaveno pro uspávání po určité době nečinnosti (která je na Windows CE dána nečinností uživatele, nikoliv systému), je nutné ošetřit uspávání při akcích trvající delší dobu. To je řešeno tak, že v pravidelných intervalech dochází k resetu systémového časovače neaktivity. Tím je systém odstaven od uspávání zařízení. Třída má v sobě automatický mechanismus, který po určité době kontroluje obsazenou, resp. volnou paměť. Pokud je několikrát po sobě vyhodnoceno, že nedochází ke změnám, je zařízení považováno za neaktivní a resety systémového časovače jsou pozastaveny. K opětovné aktivaci dochází při jakémkoli pohybu paměti. Po předem dané době tedy může (při neaktivitě systému určené neměnným poměrem obsazené a neobsazené paměti) dojít k aktivování běžného systému uspávání mobilního zařízení. Tento princip není dokonalý, nicméně pro dané účely je zcela postačující. Navíc je také možné uspávání zařízení odstavit úplně (či jej zase povolit), což se

v praxi dá výhodně použít, pokud předem víme, že akce bude značně časově náročná a nesmí dojít k uspání zařízení, jako je tomu například u synchronizace.

4.6.1.4 Framework.EventManager a Framework.TimeScheduler

EventManager je singleton reagující na předem definovaný seznam událostí určitou akcí. Seznam událostí obsahuje především systémové události jako je start či stop aplikace, načtení čárového kódu, připojení a odpojení od napájení, změna stavu CAPS locku a podobně. Pomocí konfiguračního souboru lze ke každé z těchto událostí přiřadit různé akce provedené po vzniku této události. Provedená akce je obecně volání některé služby přítomného modulu (viz odstavec 4.6.4). Vznik neboli volání těchto událostí zajišťují v závislosti na typu události jednak různé součásti frameworku a jednak moduly implementující rozhraní IEventInvoker. Takové moduly jsou EventManagerem z předem daného umístění automaticky načítány a inicializovány při startu. Využití EventManageru může být například v situaci, kdy uživatel požaduje při načtení čárového kódu zvukovou reakci zařízení. Pak je po vzniku události načtení kódu spuštěna služba modulu Beeper, která zařídí, že zařízení zapípá.

Třída CradleEvents je právě výše zmíněným modulem EventManageru (viz. Obrázek 4.7). Tato má za úkol sledovat stav připojení zařízení k napájecí síti. Po připojení či odpojení ze sítě vyvolá odpovídající událost.

TimeScheduler je třída úzce spjata s EventManagerem. Využívá jeho služeb spouštění akcí při vzniku určité události. TimeScheduler generuje na základě své konfigurace systémové události v určitém čase. Jedná se o obdobu plánovače úloh v operačních systémech Windows. V daný čas, či po uplynutí nějakého intervalu je jednou nebo periodicky spouštěna konfigurovaná událost. TimeScheduler může generovat neomezené množství událostí, které jsou vždy jednoznačně identifikovány a mohou být obslouženy EventManagerem. Takto lze například zajistit synchronizaci času, synchronizaci databáze či jiné systémové, ale i nesystémové události.

4.6.1.5 Framework.GUI

Jedná se o základní třídu pro práci s grafickým rozhraním mobilního zařízení. Obsahuje zejména jednoduché dialogy, které jsou dále použity v celém frameworku nebo jednotlivých modulech.

Třída WaitForm, jak již název napovídá, se stará o informování uživatele o stavu zaneprázdněnosti aplikace. Je to singleton umožňující zobrazení a skrytí svého formuláře.

Vzhledem k určitým omezením dialogových zpráv standardní třídou obsaženou v .NET byla implementována třída MessageBoxEx sloužící ke stejnému účelu. Obsahuje také možnosti zobrazení zprávy „na pozadí“ právě vykonávané fronty událostí, což znamená, že běh aktuálního vlákna není zobrazením zprávy zastaven.

InputBox je dialog s možností vložení uživatelského textu.

Poslední zajímavou třídou je BaseForm. Ta je předkem veškerých formulářů v celém systému a obsahuje základní nastavení pro každý formulář. Například definice konfiguračního souboru formuláře, práce s nápovědou či obsluhu standardních virtuálních kláves (viz odstavec 4.6.1.10) jako je Close, Accept, NextControl a PrevControl. Poslední zmíněné klávesy slouží pro pohyb mezi prvky ve formuláři čili simulují funkci kláves TAB a ATL+TAB. Dále obsluhuje čtečku čárového kódu a generuje patřičné události pro svého potomka, případně prvky na něm vložené. Také se stará o veškeré možnosti přetěžování virtuálních kláves na úrovni modulu či formuláře a jejich zpracování.

4.6.1.6 Framework.Gui.Components

Tato třída obsahuje základní komponenty použité při vývoji každého formuláře. Většina z nich je vytvořena pro optimalizaci chování daného prvku a také pro obsluhu standardních událostí frameworku, jako jsou například virtuální klávesy či čárové kódy. Kromě méně zajímavých komponent jako je například ButtonEx, CheckBoxEx, TextBoxEx a NumTextBoxEx, které jsou mírně modifikovanými potomky standardních prvků, jsou v této knihovně implementovány i další podstatně zajímavější třídy a prvky.

Komponenty, u nichž je takové použití rozumné, mají možnost být inicializovány šablonou. Šablona je konfigurační soubor, který danému ovládacímu prvku říká, jak se má chovat. Může být například nastaven na mód pouze pro čtení, nebo může omezit výběr hodnot pouze na některé dovolené. Šablony jsou vždy konfigurační soubory určené na jeden formulář, případně mohou být spojovány pro více formulářů.

Prvky obsahující a zobrazující data jsou standardně připraveny na binding, tj. spojení těchto dat v ovládacím prvku přímo se zdrojem dat, například v datasetu nebo jiném libovolném objektu. Většina obsažených komponent nějakým způsobem tento standardní binding ovlivňují, protože implicitní chování prvků není vyhovující.

AlphaModeKeepHelper má na starosti udržení příznaku zapnutého alpha módu. Obsahuje konstruktor, jemuž se předá ovládací prvek. Třída se pak napojí na jeho události obsluhující focus. Hlavním úkolem této třídy je u komponent vyžadující číselný vstup zrušit alpha mód a po opuštění komponenty ho (pokud nebyl v průběhu činnosti nějak změněn) zase obnovit do původního stavu. Jedná se vlastně o omezení vyplývající z klávesnic mobilních zařízení. Zařízení Motorola, ale i jiné, mají v principu numerickou klávesnici. Alpha znaky jsou vkládány, tak jako na mobilním telefonu, několikerým stisknutím daného čísla. Rozdíl zde ale je, že musí být zapnut tak zvaný ALPHA mód. Situaci, kdy do očekávaného číselného vstupu je napsán nečíselný znak, právě řeší tato třída.

Celkem jednoduchou třídou je DateTimePickerEx. Kromě standardních a výše zmíněných společných vlastností obsahuje ošetření vstupu typu NULL. Standardní prvek DateTimePicker v .NETu totiž neumožňuje s takovou hodnotou vůbec pracovat. Proto je nad běžnými vlastnostmi postavena nadstavba, která prázdné hodnoty ošetřuje jak graficky, tak potom datově při napojení přes datový binding.

DetailLabel a DetailPanel jsou komponenty určené k zobrazení formuláře obsahujícího detailní informace o zvolené položce. Data jsou pouze zobrazena, nelze je zde nikterak modifikovat. Společně s DetailFormem (viz odstavec 4.6.1.7) umožňují také načítání uživatelsky definovaného zobrazení z konfiguračního souboru. Data lze různě seskupovat a měnit vlastnosti písma zobrazených informací. Obecně jsou data uspořádána na formuláři tabulkou vždy obsahující titulek hodnoty a vlastní hodnotu.

TextBoxSelector je TextBoxEx napojený na určitý modul. Je možné do něj vkládat data a zároveň také vybrat požadovaná data z asociovaného modulu. Výběr dat ze seznamu může být spuštěn poklepáním na ikonu tří teček (pokud je v konfiguraci zapnuto její zobrazování) nebo virtuální klávesou Execute. Selector podporuje také výběr pomocí čtečky čárového kódu. Obecně je pro celý prvek určen jeden modul (například adresář). Pro výběr dat pomocí tohoto modulu je určena služba modulu. Výběr pak probíhá v seznamu všech dostupných dat daného typu z tabulky, kde mohou být zobrazeny i další doplňující údaje (například adresy, IČO a podobně). Pro výběr dat scannerem je určena další služba. Vyhledávání ruční a vyhledávání čtečkou může tedy probíhat podle odlišných kritérií. Více o modulech a jejich službách je popsáno v části 4.6.4.

ComboBoxSelector se podobá TextBoxSelectoru s tím rozdílem, že umožňuje přímý výběr dat z rozbaleného seznamu. Může být klasicky typu DropDown či DropDownList, který omezuje vybrané hodnoty pouze na hodnoty obsažené ve zdroji dat. Opět je možné k identifikaci použít scanner nebo úplné rozbalení celého seznamu. Pro zobrazení seznamu položek přímo v ComboBoxu je použita speciálně definovaná služba přiřazeného modulu, tak jako je tomu v případě čtečky čárových kódů.

Poslední komponentou je DataGridEx. Jelikož standardní DataGrid z Compact Frameworku je velice omezen, bylo nutné implementovat hodně funkcionality do zděděné komponenty a následně pak také i do celého ovládacího prvku využívající tuto komponentu. Bylo nutné ošetřit například velikosti scrollbarů, jelikož tato komponenta je má napevno stanovené a nemění se automaticky se systémovým nastavením jako u ostatních prvků. Dále pak ošetření virtuálních kláves a identifikací jednoho vybraného řádku v tabulce a přidání událostí na kliknutí a dvojité kliknutí na různé oblasti v tabulce jako je například hlavička sloupce nebo podbarvování vybraného řádku a sloupce. V neposlední řadě je zde ošetřeno obnovení dat z přiřazeného datového zdroje, které ve standardní komponentě nepracuje úplně dokonale a navíc se projevuje nepříjemným blikáním uživatelského rozhraní.

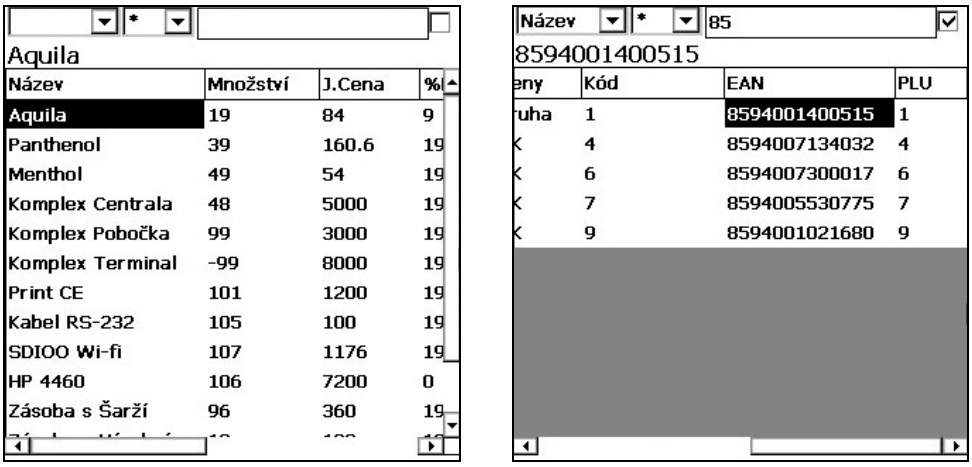
4.6.1.7 Framework.GUI.Forms a Framework.GUI.UserControls

Tyto knihovny obsahují formuláře a ovládací prvky použité při práci s grafickým uživatelským rozhraním ve frameworku a modulech. Jedním z nich je DetailForm, který se stará o zobrazení detailů určité vybrané položky. Umožňuje navíc uživatelskou konfiguraci zobrazení. Nejdůležitějším prvkem v této skupině je ale obalení nad komponentou DataGridEx.

Panel s DataGridem obsahuje několik prvků. Jsou to především komponenty určené pro filtrování a vyhledávání v tabulce, ale také místo pro zobrazení zpráv a obsahu právě vybraného pole (status). Filtrovací prvky sestávají z pole pro výběr filtrovaného sloupce, porovnávacího operátoru a vlastní hodnoty filtru. Dále pak checkbox pro povolení a zakázání daného filtru. V status panelu je vždy zobrazen detail vybraného pole. Je to proto, že v buňce v tabulce nemusí být vždy celý obsah viditelný. Dále zde pak jsou zobrazeny informace o počtu řádků a aktuálním vybraném a je použit jako výstup pro agregační funkce typu minimum, maximum, průměr a suma. Celá komponenta funguje se třemi typy dat. Jsou to data statická, tedy přímé přiřazení DataTable. Data dynamická, tj. přiřazení SQL dotazu a následně pak data kombinovaná, což umožňuje od dynamických dat (SQL) odečíst data statická (DataTable) a zobrazit tak jejich rozdíl. Ve všech těchto typech funguje vyhledávání i ostatní vlastnosti. Dále je možné u každého gridu pomocí konfiguračního souboru určit, které sloupce jsou viditelné a podle kterého sloupce se data řadí. To je následně možné přetížit kliknutím na jméno sloupce. Při přiřazení dat dynamicky, tedy SQL dotazem, může být zapnuta funkce LoadOnDemand, která aktivuje zpožděné načítání dat na požádání. Funkce je implementována zejména kvůli paměťovým a výpočetním omezením mobilního zařízení, jež nemusí být schopno v rozumné časové době načíst a pojmout celý dotaz reprezentovaný tabulkou. Funkce automaticky donačítává další data v druhém vlákně, jakmile se uživatel přiblíží ke konci již načteného seznamu. Ideálně jsou pak další data dostupná dříve, než je uživatel potřebuje.

V gridu je možno vyhledávat pomocí čtečky čárového kódu. Komponenta se k ní chová jako by to byl standardní uživatelský vstup z klávesnice a je vyhledáváno v aktuálně označeném sloupci. DataGrid dále umožňuje vývojáři zadat pevný filtr, který je aplikován při každém vyhledávání dat. Jeho použití je velice výhodné například při vyhledávání v položkách dokladu, kdy v databázi existuje dokladů více. Pak je tabulka takto omezena na jediný konkrétní doklad.

DataGrid poskytuje vývojáři několik dalších příznaků a metod pro práci s načtenými daty, zejména informace týkající se identifikace aktuálního řádku a získání dat z něj.



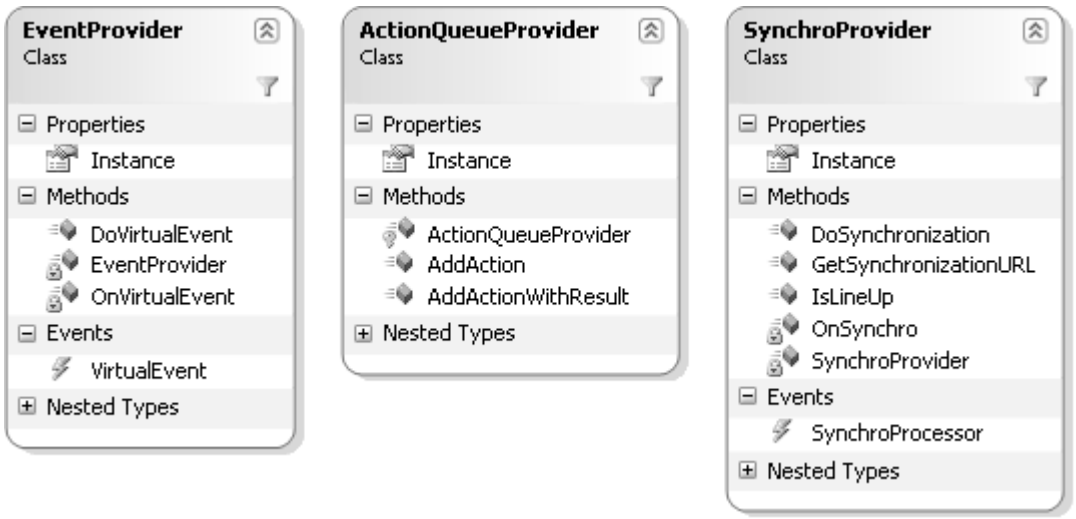
Obrázek 4.5 - ukázky DataGridEx

4.6.1.8 Framework.Interfaces

Tato knihovna obsahuje rozhraní použitá vně frameworku a bývá z ní přímo čerpáno v modulech a komponentách využívajících služeb frameworku. Mimo rozhraní obsahuje i definice několika typů s rozhraními úzce souvisejícími. Je zde například rozhraní, které musí implementovat všechny moduly, s nimiž pak framework pracuje. Dále pak jsou zde pomocná rozhraní pro práci s komponentami GUI a také důležitá třída ObjectIdentifier. Ta obsahuje kromě vlastností pro uživatelské zobrazení hlavně datový řádek a ID aktuálně vybraného záznamu. Slouží tedy pro předávání vybrané položky například z datagridu nebo mezi jednotlivými moduly.

4.6.1.9 Framework.InterOp

Knihovna InterOp slouží ke komunikaci mezi frameworkem a modulem či aplikací. Je to jediné veřejné místo, pomocí něhož mohou externí procesy (což jsou z principu i moduly) zasahovat do frameworku a posílat mu různé zprávy. Také je to místo, které poskytuje služby frameworku a většina tříd tento fakt nese ve svém názvu, jedná se o takzvané providery. Jejich struktura je naznačena v následujícím diagramu.



Obrázek 4.6 – ukázky některých providerů

ActionQueueProvider je singleton pomocí něhož může uživatel (uživatelem pro tuto třídu je jiná třída, nikoli osoba) vložit do fronty událostí nějakou metodu, která bude zpracována později. Znamená to, že se nezastaví běh aktuálního vlákna vkládajícího danou metodu. Systém opožděného zpracování funguje na principu událostí systému Windows generovaných při vložení požadované metody. Obsluha takto generované události nastane až při jejím přijetí, nikoli okamžitě. Metody jsou řazeny do fronty, mohou být volány i s parametry a mohou vracet výsledek. Ten je vrácen pomocí zpětného volání metody předané při inicializaci celé akce opožděného spouštění.

Pro obsluhu čárových kódů je implementována třída BarcodeScannerProvider. Poskytuje místo pro registraci posluchačů, kteří chtějí čárový kód po načtení získat. Je zde tedy fronta posluchačů, z nichž první, který zpracuje čárový kód, ruší volání dalších. Tato třída slouží také k vytvoření události načtení čárového kódu pro externí moduly obsluhující toto zařízení.

EventProvider umožňuje modulům a frameworku vyvolávat virtuální události. Jsou to stejné události, které jsou zpracovávány EventManagerem (viz. odstavec 4.6.1.4). Zde na jejich výskyt mohou být ovšem registrovány i jiné programové komponenty a moduly. EventManager slouží pouze k spuštění uživatelské reakce na vyvolanou událost.

SynchroProvider poskytuje pro okolní svět přístup k synchronizačním službám frameworku. Je tedy úzce spojen s příslušnou částí celého frameworku (viz odstavec 4.6.1.2).

Poměrně komplikovanou třídou s návazností na další část (4.6.1.10) je KeyboardProvider. Její chování a činnost budou vysvětleny v následující kapitole.

Poslední třídou poskytující služby frameworku je PackageProcessor. Jedná se o třídu pracující s balíčky. Je to tedy třída používaná zejména při synchronizaci a nebo přípravě na ní. Obsahuje metody pro zapracování XML balíčku do databáze (vytvoření standardní databázové struktury), uložení XML balíčku do databáze, vytvoření XML balíčku ze standardní databázové struktury (tj. z tabulek v databázi) a mazání balíčků a dat z databáze. Více o práci s databází bude popsáno v odstavci 4.6.2.

4.6.1.10 Framework.Keyboard

Součást frameworku nazvaná keyboard se pochopitelně stará o práci s klávesnicí. Vzhledem k tomu, že každé mobilní zařízení má jinou klávesnici, byla mezi hardwarovou klávesnici a software implementována mezivrstva, jež tyto dvě části spojuje. Rozdíly mezi klávesnicemi mohou být opravdu diametrální. Například na datovém terminálu MC1000 jsou poměrně snadno dostupné šipky a funkční klávesy jako je TAB, F1-F12, ovšem pokud se podíváme na zařízení MC70, zjistíme, že zde to s šipkami již tak jednoduché není. Pro stisknutí šipek vlevo a vpravo je nutné přepínat klávesnici do ALPHA módu. Dalším kladným důsledkem oddělení HW klávesnice od frameworku je fakt, že každý uživatel má jiné požadavky na obsluhu mobilního zařízení a tato vrstva může odlišným požadavkům uživatelů vyhovět. Takzvaná vrstva virtuálních kláves je totiž snadno konfigurovatelná pomocí XML souboru.

Ve zkratce jsou v této vrstvě definovány určité virtuální klávesy (například Accept, Close, Help, NextControl atd.) a na fyzické klávesy (tedy například šipka vlevo, 2, a, TAB apod.) jsou tyto virtuální klávesy namapovány. Vzhledem k omezenému počtu fyzických kláves na klávesnici zařízení je navíc pracováno i s časem, tedy dobou držení klávesy. V praxi to znamená, že například klávesa ESC může být namapována na virtuální klávesu RESET pro krátký stisk a pro stisk nad 1s je napárována na virtuální klávesu CLOSE. Takové ovládání se může zdát uživateli ze začátku matoucí, nicméně po nějakém čase určitě zmiňovanou vlastnost ocení.

Framework je pomocí hook metod připojen na rozhraní systému Windows a zachytává veškeré stisknuté klávesy. V případě, že vyhodnotí klávesu jako svoji (tj. má definovanou akci spojenou s touto klávesou), posílá událost stisku virtuální klávesy dále do vnitřní struktury. V opačném případě je klávesa posílána dále do systému, který ji následně zpracuje standardně.

Celý princip je navržen tak, že ke každé virtuální klávese může být přiřazeno více kláves fyzických s různou kombinací časového stisknutí. Systém má v sobě také poměrně důmyslnou logiku práce se stisky kláves odlišenými časem. Pokud lze danou virtuální klávesu určit okamžitě, nikdy se nečeká na uvolnění klávesy a akce je vykonána ihned.

KeyboardProvider ze sekce InterOp slouží právě jako provider akcí stisku virtuální klávesy. Veškeré komponenty obsluhující stisk klávesy jsou napojeny na něj. Týká se to například předka pro všechny formuláře, který tímto reaguje na klávesy jako je Close či Help. V tomto předkovi lze také načíst doplňující konfiguraci virtuálních kláves, která může mapování přetížit, zrušit či doplnit přímo na každý formulář. Stejná funkce přetěžování je implementována i na úrovni celých modulů.

Virtuální klávesy jsou ale také používány ve velké většině ovládacích prvků, jež mají definovány své vlastní akce na obvyklé klávesy. Vzhledem k tomu, že celý systém běží na Compact Frameworku, jehož komponenty nejsou samy o sobě tak dobře ovladatelné jako na plném Frameworku, a ovládání musí být jednoduché s použitím pouhé klávesnice, je nutné upravit či doplnit standardní chování velké většiny klasických ovládacích prvků.

Pomocí vrstvy virtuálních kláves lze třeba definovat, že procházení mezi ovládacími prvky se kromě standardních kláves TAB aktivuje také šípkou nahoru a dolu. Toto chování je výhodné u zařízení bez klávesy TAB, případně tam, kde je tato klávesa špatně dostupná.

Kromě virtuální klávesy je na stisk fyzické klávesy možné přiřadit také spouštění externí aplikace. Může takto být používána například kalkulačka nebo jiná užitečná aplikace, která není součástí frameworku.

4.6.1.11 Framework.Menu

Knihovna pro práci s menu je vlastně hlavní spouštěnou částí celého systému. Je to první formulář, který je zobrazen uživateli a z něj jsou následně spouštěny veškeré další uživatelské akce. Menu je načítáno z konfiguračního souboru. Obecně může obsahovat složitější stromovou strukturu, nicméně na konci je vždy spuštěn určitý modul, resp. jeho služba. Více o modulech a službách v sekci 4.6.4. Položky v každém menu jsou číslovány a může jich být maximálně 9. Knihovna zajišťuje celkovou navigaci po menu a tím také navigaci v konfiguračním souboru menu. Položky v menu mohou být chráněny přístupovým heslem kódovaným MD5 hashí.

4.6.1.12 Framework.Modules

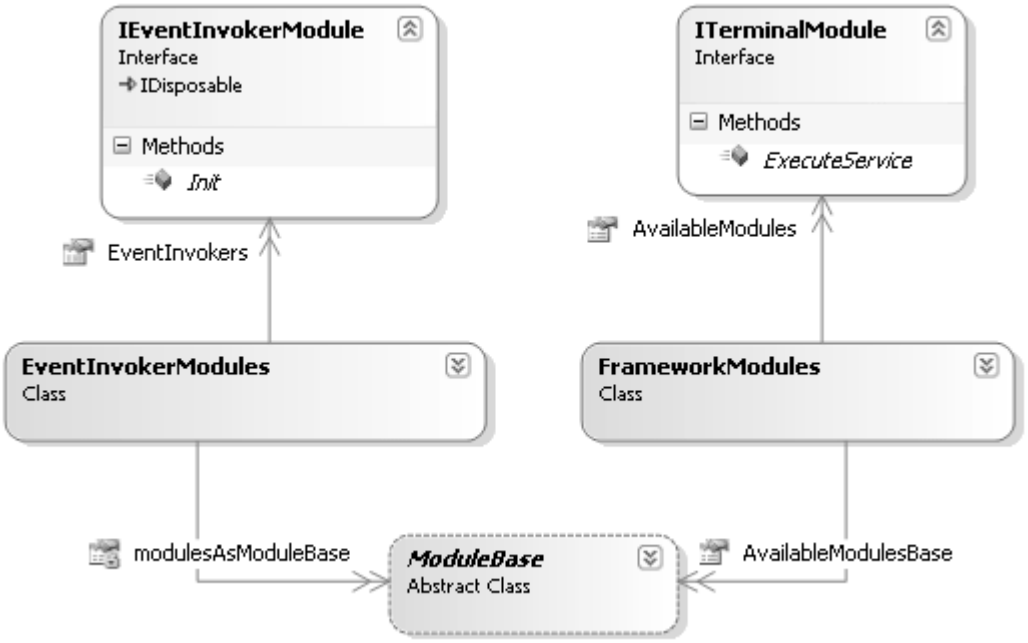
Tato část frameworku se stará obecně o načítání modulů a přístup k nim. Kromě pomocné třídy pro práci s modulární architekturou obsahuje dva seznamy modulů.

Jsou to hlavně moduly typu FrameworkModule. To jsou klasické moduly, které mohou být spouštěny a poskytují ostatním modulům nebo uživateli určité služby.

Druhou kategorií jsou moduly typu EventInvoker. Ty obsahují všechny moduly, jež mají za úkol vyvolávat virtuální události. Může to být například modul pro práci se čtečkou čárového kódu.

Všechny tyto moduly jsou načítány automaticky při startu. Každý modul může mít svou konfiguraci nezávislou na frameworku.

Z tohoto umístění mohou také moduly získávat odkazy na jiné moduly a využívat tak služeb jiných modulů. Například založení dokladu bude zřejmě využívat služeb modulu adresáře, který mu musí umožnit identifikovat a vložit do dokladu zákazníka.



Obrázek 4.7 – diagram tříd singletonů obsluhujících moduly

4.6.2 Databáze

Databáze v mobilním zařízení sestává z jednoho souboru s koncovkou *.sdf. Jedná se o databázi SQL Compact. Tento soubor je možné přenášet a otvírat na jakémkoli jiném zařízení či počítači, který disponuje potřebným aplikačním vybavením pro práci s tímto typem databáze. Na mobilním zařízení je přístup zajištěn pomocí DLL knihoven od výrobce databáze, tedy Microsoftu. Díky tomu není třeba provádět žádnou instalaci, knihovny stačí nakopírovat do systému či adresáře příslušné aplikace.

Jelikož s daty je v některých částech systému zacházeno jako s XML balíčky a jinde je k nim zase přistupováno přímo na databázové úrovni, jsou implementovány metody pro převod mezi těmito dvěma stavy. Převod XML souboru do databázových tabulek a naopak lze provádět před DataSet,

který v .NET frameworku obsahuje metody pro práci s XML. Před zapracováním do databáze je ale nutné u balíčku ještě provést pár úprav.

V databázi se musí založit příslušné tabulky, což dle struktury datasetu není problém. Zajímavější je ale práce s indexy. Automaticky lze zakládat pouze indexy na primárním klíči v každé tabulce. Další indexy musí být definovány explicitně. Třídy mající na starost zakládání datových tabulek jsou implementovány tak, že se při založení podívají do konfiguračního souboru, kde mohou být přídavné indexy pro jakoukoli tabulku definovány. Soubor je plně uživatelsky konfigurovatelný, což umožňuje indexy definovat opravdu dynamicky. Pokud zákazník často využívá řazení dle určitého sloupce, lze tímto způsobem na daný sloupec vytvořit index.

V případě, že je tabulka již založena, stále ještě není možné nový balíček do databáze vložit. Před samotným vložením je nutné přečíslovat primární indexy všech tabulek tak, aby byla data později identifikovatelná. Například každý doklad musí mít své ID.

Při převodu na druhou stranu, to znamená z databáze do XML balíčku, je situace jednodušší. Stačí příslušná data načíst opět do datasetu a následně uložit do XML. K tomuto výstupu je ještě přidána hlavička doprovázející data, která obsahuje identifikaci typu dat a datum vytvoření balíčku, a balíček je kompletní. V mobilním zařízení jsou XML balíčky ukládány také do SQL databáze, do speciální tabulky obsahující všechny balíčky. Obecně všechny moduly pracují s daty v podobě databázových tabulek a až před odesláním jsou tato data převedena do balíčku. Jakmile jsou takto převedena již nelze data modifikovat ani s nimi nikterak pracovat. Znamená to například, že veškeré tisky, vytváření navazujících dokladů a podobně musí být provedeny před vytvořením XML balíčku.

4.6.3 Synchronizace mobilního zařízení

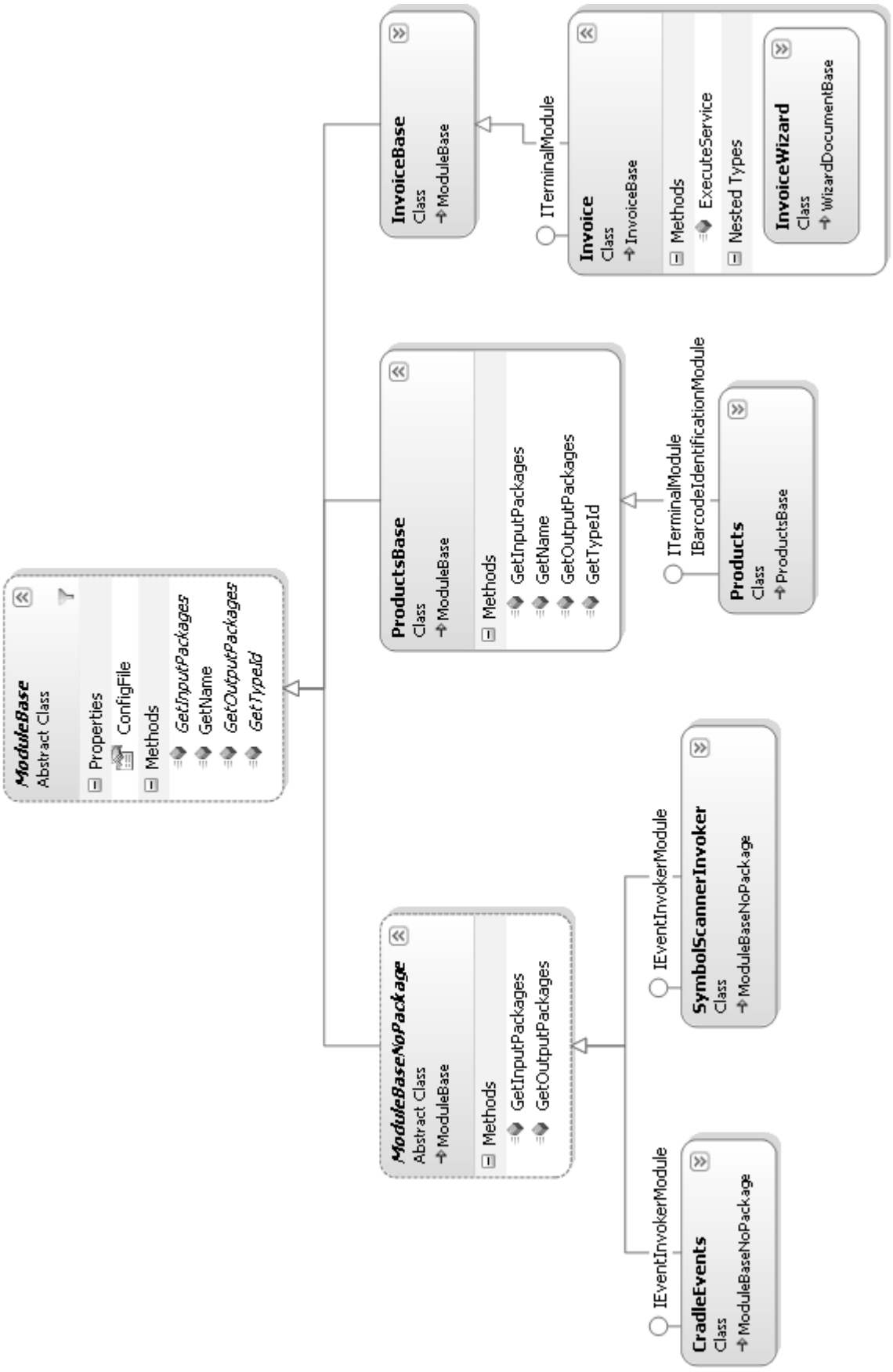
Synchronizace probíhá pomocí posílání balíčků přes webovou synchronizační službu na straně koncového systému. Při synchronizaci vstupních balíčků do terminálu je použito metody částečného nahrávání souboru. Tato metoda je použita pro balíčky přesahující definovanou velikost (v současné době 1MB). V průběhu komunikace je použito ukládání do dočasného souboru. Po kompletním stažení balíčku je na mobilním zařízení jeho obsah uložen do databáze, tato procedura může v závislosti na velikosti dat a rychlosti zařízení trvat i několik minut.

4.6.4 Moduly

Po startu aplikace na mobilním zařízení jsou inicializovány všechny moduly. Každý modul může mít v podadresáři conf konfigurační soubor. Moduly obecně poskytují svým klientům služby. Klientem může být uživatel, ale také například další modul. Služba je v modulu identifikována číslem a mohou jí být předány libovolné parametry. Framework sám o sobě nevykonává žádnou činnost, pouze poskytuje modulům prostředky pro práci. Moduly načítané frameworkem musí vždy implementovat rozhraní ITerminalModule.

Standardní činnost systému na mobilním zařízení vypadá tak, že jsou postupně po sobě volány různé moduly. Ať už se jedná o výkonné moduly či moduly pomocné (systémové). Pokud je tedy například vytvářena nová příjemka je nejprve spuštěn modul příjemka se službou vytvoření nové. V rámci tohoto modulu jsou volány služby dalších modulů (identifikace zákazníka, identifikace produktů) a následně je doklad předán (resp. jeho ID) modulu tisk příjemky. Po vytisknutí dokladu může být tento opět předán třeba do synchronizace. Toto chování je samozřejmě plně uživatelsky nastavitelné pomocí konfigurací menu a příslušných modulů. Obecně je ale výstup předchozího modulu předáván do modulu následujícího a na tomto základu pracuje celý systém.

Jsou to tedy právě moduly, kdo vytváří data. Jejich dělení do několika kategorií je popsáno v následujících kapitolách.



Obrázek 4.8 – diagram tříd základních typů modulů

4.6.4.1 Systémové moduly

Systémové moduly jsou ty, jež nemají přímou vazbu na žádný hostitelský systém ani na konkrétní mobilní zařízení. Je to v podstatě nadstavba frameworku. Poskytují totiž pro další moduly určité služby. Tato kategorie modulů většinou není závislá na ostatních funkčních částech frameworku (samozřejmě je závislá na grafických funkcích).

ApplicationRunner spouští libovolné procesy. Je možné mu předat cestu a soubor, který má spustit, a on jej spustí. Používá se hlavně v menu či ve spojení s definovanou virtuální klávesou, například pro spouštění utilit jako je kalkulačka.

ComposeModule je nejsložitější ze systémových modulů co se konfigurace týče, jeho princip je ale jednoduchý. Umožňuje volat za sebou sekvenčně více služeb rozdílných modulů. Standardně je totiž v jednom místě možné volat pouze jeden modul, resp. jednu službu volaného modulu. Konfigurace tohoto modulu může být tedy opravdu složitá a hlavně rozsáhlá, tento modul je využíván v podstatě při každé akci. Dále je do sekvence po sobě jdoucích akcí zakomponována i možnost volby, která je vyhodnocována na základě uživatelského vstupu. Je tedy možné uživateli zobrazit dialog s otázkou, co chce dělat dále. Dle toho potom ComposeModule umí běh programu větvit.

DocumentNumbers se zabývá číselnými řadami. Pod jednoznačným identifikátorem jsou zde ukládány různé číselné řady. Ty mají definován svůj prefix, postfix, celkovou délku a aktuální hodnotu. Modul obsahuje služby pro zvýšení či snížení čísla v číselné řadě. Každý doklad pracuje se svojí číselnou řadou nezávislou na ostatních dokladech.

GlobalSettings je modul sloužící jako datový sklad. V podstatě do něj mohou moduly ukládat jakákoli data, ale většinou se používá pro uložení různých konfigurací a nastavení. Hodnoty jsou uloženy v XML souboru. Každá hodnota má svůj identifikátor pomocí něhož ji lze kdykoli opětovně načíst.

PrinterSettings obsahuje definice tiskových profilů. Tento modul je úzce spojen s knihovnou PrinterCE, pro niž profily spravuje. Je zde možné definovat sady nastavení tiskáren, které se pak při tisku dají používat a uživatel není dotazován na neustále se opakující nastavení.

ServiceMessage je modul, který slouží pro vytváření systémových zpráv. Typ zprávy je zadán jako parametr při spouštění modulu, hodnota zprávy může být zadána stejně tak a nebo se na ni modul umí zeptat přímo uživatele. Formát zprávy má vždy pouze tyto dva parametry a to je typ a vlastní hodnotu. Oba jsou typu řetězec. Servisní zprávy jsou obecně definovaná struktura, která je obvykle zpracovávána až na úrovni BackEnd Frameworku příslušnou akcí. Jedná se zejména o uživatelsky modifikované chování některých procesů, například zpracovávání dokladů jiným způsobem.

Synchronization se stará o posílání a příjem balíčků mezi mobilním zařízením a synchronizačním serverem. Obsahuje služby pro příjem nových balíčků a odeslání stávajících. To lze provádět pro jednotlivé balíčky dle předaného ID a nebo pro všechny balíčky daného typu. Navíc právě tento modul obstarává pro uživatele službu vytváření XML balíčků z databázových tabulek.

Modul synchronizace je závislý na frameworku, který poskytuje služby vlastní komunikace se vzdáleným úložištěm.

Modul TimeSync zajišťuje synchronizaci systémových hodin mobilního zařízení s externím serverem. Tím může být přímo synchronizační server a nebo také nějaká internetová stránka. Aktuální datum a čas jsou čteny z http hlavičky zadané stránky a nastaveny jako aktuální čas. Pro práci se systémovým časem je opět nutné používat nativní funkce Windows CE. Tento princip synchronizace samozřejmě kvůli zpožděním při přenosu stránky od serveru až k mobilnímu zařízení není příliš přesný, ale pro účely mobilního zařízení je tato přesnost dostačující.

Dále pak jsou zde další pomocné moduly, jejichž funkce ale nejsou až tak zajímavé a slouží především systémovým úlohám frameworku například pro zobrazení hlášení, výpočtu MD5 hashe atd.

4.6.4.2 Symbol moduly

Kategorie modulů Symbol je pojmenována dle výrobce zařízení MC1000 a vyšších. Logicky je tedy veškerá funkcionalita těchto modulů spjata jen a pouze s těmito zařízeními a u zařízení jiných výrobců je použít nelze. Jedná se o jednoduché moduly pro obsluhu klávesnice, indikačních zařízení a také čtečky čárových kódů. Tyto moduly bývají závislé především na SDK knihovnách výrobce a také na InterOp části Terminal frameworku. Pomocí této části komunikují moduly, pokud je to zapotřebí, s ostatními částmi systému. Například tak posílá modul čtečky čárových kódů snímaný kód.

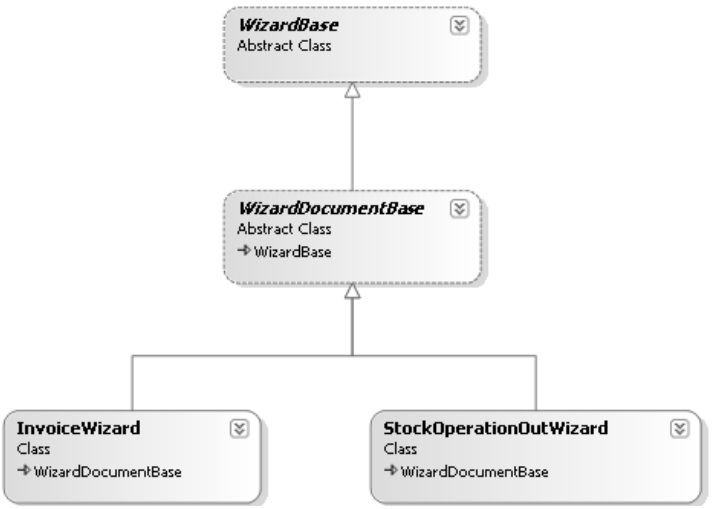
4.6.4.3 Pohoda moduly

Pohoda moduly jsou z hlediska celé aplikace s napojením na hostitelský systém Pohoda nejdůležitější částí. Jedná se totiž o soubor výkonných metod, pomocí nichž se pracuje s daty. Tyto moduly tedy vytváří doklady na základě vstupu uživatele a číselníků hostitelského systému.

Zatímco ostatní moduly pracují na bázi poskytnutí jednoduchých služeb, případně zobrazení jednoho dialogu očekávajícího vstup, některé z této kategorie modulů jsou poněkud složitější. Jedná se o moduly v režimu průvodce. Průvodce proto, že při práci s tímto modulem je nutné procházet přes více dialogů. Například při vytváření jednoduchých dokladů je to hlavička, dále pak položky a sumář vložených dat. K práci nad modulem typu průvodce slouží různé pomocné třídy. Jako obal nad celým procesem je implementována abstraktní třída WizardBase, která postupně inicializuje všechny formuláře a data a následně se stará o jejich postupné zobrazení a předávání informací mezi nimi.

Pro práci se standardními doklady byla vytvořena zděděná třída s názvem WizardDocumentBase. Slovo document je zde klíčové. Jedná se totiž o potomka WizardBase, který umí pracovat se standardním dokumentem. Tím se rozumí doklad obsahující běžné údaje jako je hlavička dokladu a položky. Má přidělenu svou číselnou řadu a obsahuje běžné formuláře pro editaci

dokladu. Tato třída je opět abstraktní a obsahuje hodně virtuálních metod, jelikož tohle je právě základní kámen každého modulu a musí umožňovat jemné změny způsobené odlišnostmi jednotlivých dokumentů. Jeho základní úlohou je postarat se o založení a ukládání dokumentu. Při vývoji nového modulu je to tedy tato třída, které je zděděna.



Obrázek 4.9 – diagram tříd průvodce

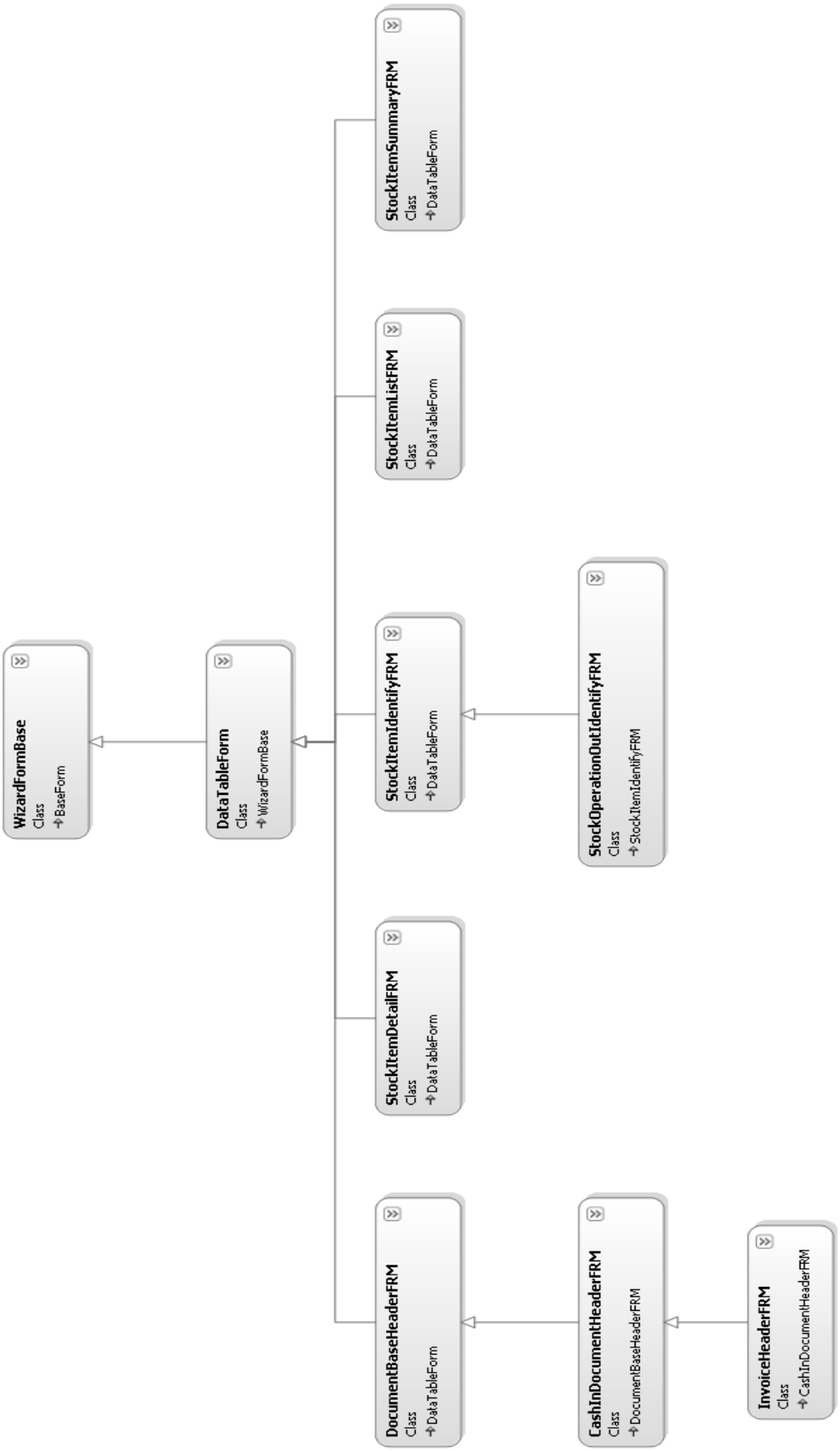
Formuláře obsažené v průvodci jsou potomky speciálního typu WizardFormBase. Ten kromě méně zajímavých věcí obsahuje dvě vlastnosti, jedna je WizardMoveResult a druhá WizardActionResult. Podle nich určuje WizardBase jak se má chovat po ukončení aktuálního formuláře. Pokud se má uživateli zobrazit například následující formulář, je nutné, aby ve WizardMoveResult bylo nastaveno „pohyb ve formulářích o jeden vpřed“. Analogicky při jiných pohybech. V případě, že má být provedena i nějaká akce (například smazání aktuálního záznamu, vytvoření nového, pohyb mezi záznamy) je nastavena příslušná akce do WizardActionResult. Dále má každý formulář svůj vstupní a výstupní objekt, pomocí nichž jsou předávána zpracovávaná data. Ta jsou automaticky při pohybu předána z výstupu aktuálního na vstup nového formuláře. Definice toho jak se mají jednotlivé formuláře chovat, kdy se například posunout kupředu či zpět a při jaké události, je obsažena v konfiguraci. Pro každý formulář je definován soubor akcí a pohybů, které nastanou po stisku určité virtuální klávesy. V hlavičce dokladu pak může být například nastaveno, že na stisk virtuální klávesy SAVE má provést uložení aktuálního dokladu a navíc se posunout o jeden formulář dopředu, což je většinou v tomto případě vkládání položek do dokladu.

Speciálním typem průvodcovského formuláře je DataTableForm. Je použit pokaždé, když je daný formulář přímo spojen s nějakými daty. Obsahuje odkaz na DataTable a informaci v jakém řádku tabulky se uživatel právě nachází. Jsou zde tedy konkrétně definována data, na něž bude formulář napárován. Také obsahuje akce pro spojení dat s uživatelskými ovládacími prvky (databinding).

Dále pak jsou implementovány formuláře reflektující nejběžnější potřeby každého dokladu. Je to formulář pro hlavičku dokladu, seznam vložených produktů, detail produktu, formulář pro vkládání produktů do dokladu a nakonec zobrazení celkového přehledu o dokladu. Tyto třídy jsou opět různě děděny pro konkrétní doklad tak, aby postihly veškerá specifika (viz. Obrázek 4.10).

Jsou zde dva základní druhy modulů. Jedny jsou moduly pro práci s daty vstupujícími do mobilního zařízení z hostitelského systému. To jsou například produkty a adresář. Tyto moduly umožňují hlavně identifikaci daného záznamu a dále pak zobrazení detailů o záznamu a další drobnosti. Ty běžně nepracují v režimu průvodce.

Druhým typem jsou moduly vytvářející nová data, což jsou vlastní doklady. Tedy výstup celého systému. Ty umožňují většinou to samé pro každý dokument a to vytvoření, editaci a smazání. Zástupci modulů v této kategorii jsou faktura, prodejka, příjemka, převodka a výdejka. Speciálním typem dokumentu je pak inventura, která se ale do Pohody importuje jako soubor dokumentů příjemky a výdejky. Každý doklad má svá určitá specifika, avšak v principu jsou všechny doklady podobné. Obsahují hlavičku a dále pak jednotlivé položky. Standardní údaje v hlavičce jsou například datum, text dokladu či obchodní partner. Položky vkládané do dokladů mohou být buď textové a nebo s vazbou do skladu (tedy je vložen konkrétní produkt). Většina dokladů je až na pár odlišností stejná, poněkud jiný systém práce je u inventury. Zde nejsou vkládány položky ze skladu, ale z předem nachystané dočasné tabulky vytvořené při založení inventury. Dále je pak drobný ale podstatný rozdíl u příjemky, kde jsou skladové operace místo výdejových chápány jako příjmové a u převodky, kde skladové operace jsou zároveň výstupní z jednoho produktu a vstupní do produktu jiného (myšleno co se množství na skladě týče). Všechny doklady fungují tak, že po editaci hlavičky je možné začít do dokladu vkládat zásoby. Buď textově a nebo výběrem ze skladu. V případě, že je zásoba nutně určena dalším bližším identifikátorem (jako například sériové číslo), musí být vybrána přímo konkrétní zásoba. Po vložení všech položek je doklad uložen a tím i aktualizováno množství použitých zásob na skladě.



Obrázek 4.10 – diagram tříd formulářů včetně ukázky dědění u faktury a výdejky

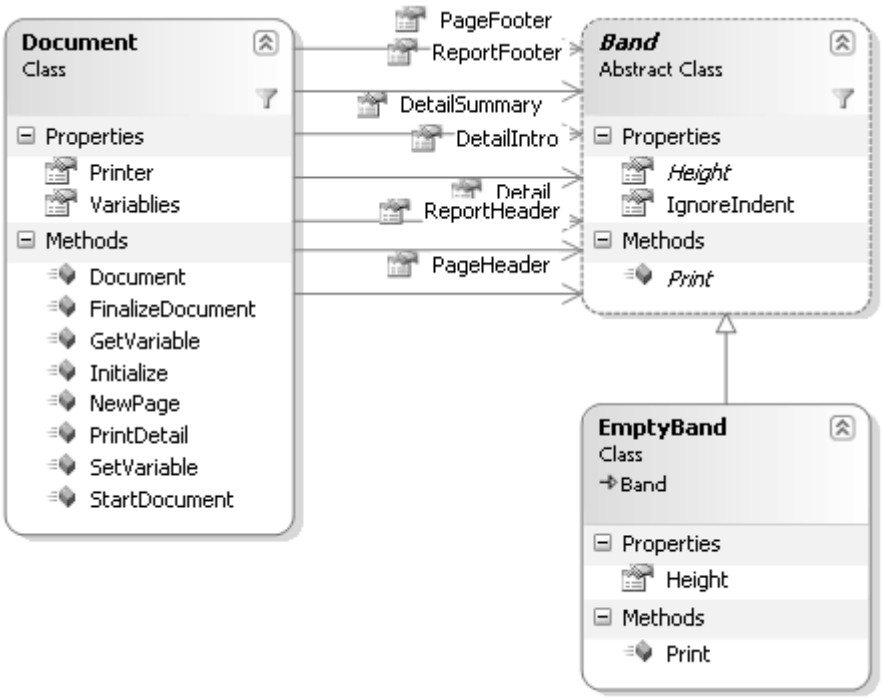
4.6.4.4 Tiskové moduly

Tiskové výstupy z mobilního zařízení jsou řešeny speciální třídou modulů. Není to tedy tak, že by přímo každý modul pro tvorbu dokladu uměl tento doklad i tisknout. Pokud daný doklad tisk vyžaduje, je pro něj vytvořen nový tiskový modul.

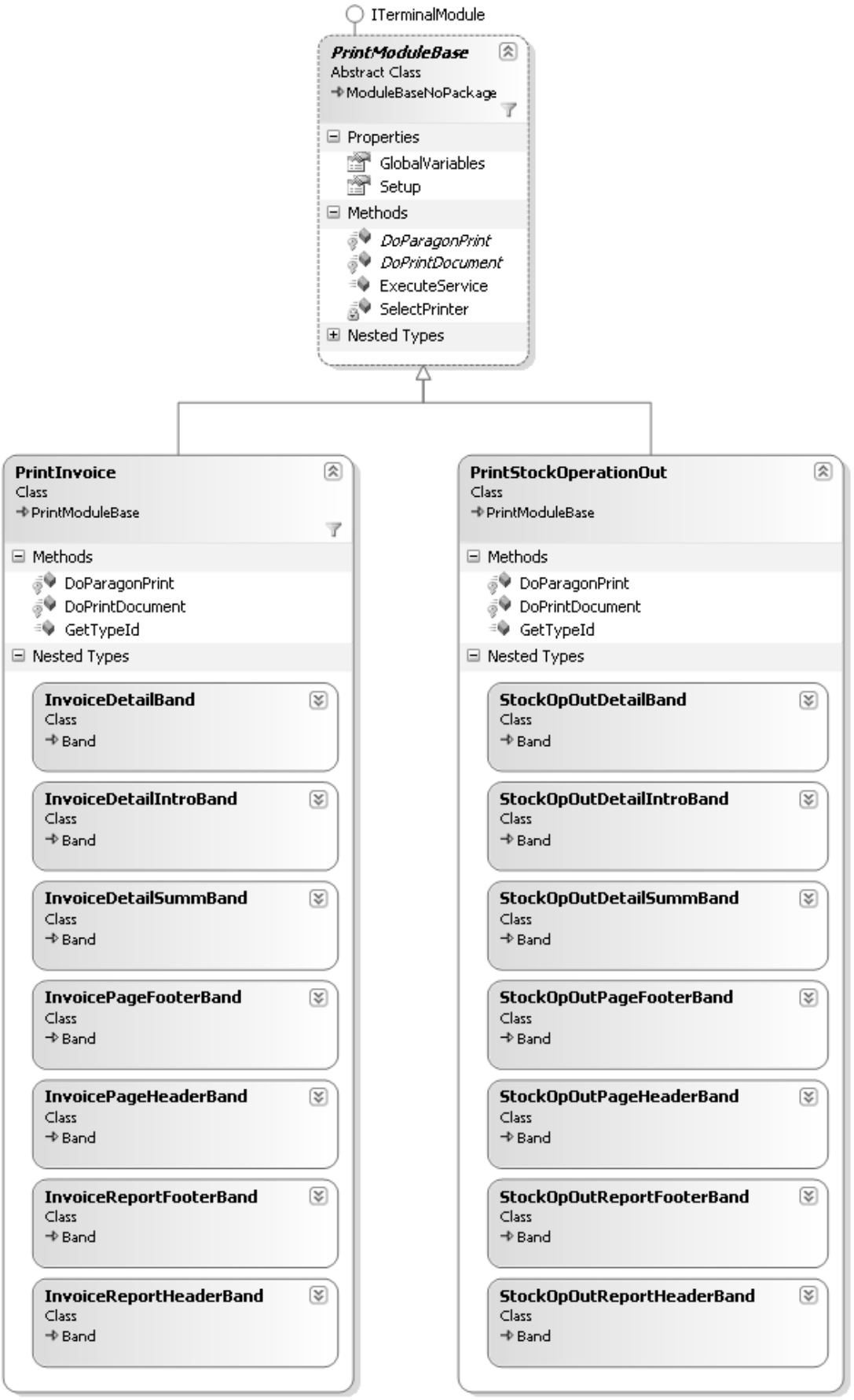
Pro veškeré tisky bylo vytvořeno několik pomocných tříd pro načítání dat z databáze, ale také pro vlastní tisk. Například výběr tiskárny popsaný výše. V každém modulu může být více tiskových sestav, které jsou buď pro A4 tiskárnu nebo tiskárnu paragonovou. Klasická tisková sestava je vytvořena pomocí základních funkcí pro práci s tiskem jako je tvoření čar a tisk textu v různých fontech. Paragonový tisk je prováděn přímým posíláním textu na příslušnou tiskárnu. Jde o tak zvaný ASCII tisk. O připojení k tiskárně a realizaci protokolu se vždy stará tisková knihovna PrinterCE.Net.

Pro tiskové moduly existuje předek (viz. Obrázek 4.12), kterého musí každý modul dědit a zároveň obsahuje metody pro základní práci s tiskárnou. Vývoj nového tiskového modulu je tedy otázkou implementace potomka této třídy a řeší se pouze skutečné rozložení prvků na sestavě a načtení daných hodnot z databáze.

U tisku na A4 tiskárnu je navíc implementována třída (viz. Obrázek 4.11), která obsahuje celou tiskovou sestavu. Umožňuje definovat tisky na úrovni jednotlivých „oddělení“ (band) dat jako je záhlaví stránky, záhlaví sestavy, položky, zápatí stránky a podobně. Po této definici je pak automaticky řešeno i stránkování sestavy a po odstránkování jsou opět na novou stránku vytištěna příslušná data (např. záhlaví stránky). Většina standardních sestav tedy používá služeb této třídy a pouze nastavuje vzhled jednotlivých tiskových oddělení.



Obrázek 4.11 – základní tiskové třídy



Obrázek 4.12 – ukázka implementace tiskových modulů

5 Implementace

Implementace celého systému probíhala až na pár drobností poměrně hladce. Samozřejmě došlo, jak tomu bývá u většiny projektů podobného typu, k různým odlišnostem od návrhu i úpravám kvůli pozdějším změnám v zadání. Veškerý kód je psán v jazyce C# a tedy pro jeho tvorbu bylo využito vývojové prostředí Visual Studio ve verzi 2005. Společně s používáním verzovacího nástroje a systému pro evidenci chyb vzniklých při vývoji bylo velice solidním základem pro celou práci.

Při implementaci všech součástí celého systému byl kladen důraz zejména na rozšiřitelnost a škálovatelnost aplikace. Systém je tedy velice modulární a poměrně dobře upravovatelný. Většina tříd je od počátku navržena tak, aby v případě potřeby mohla být zděděna, případně doplněna o další funkce použitím externích modulů. Úpravy dle konkrétních potřeb zákazníka jsou totiž v této oblasti očekávány. Systém je vyvíjen pro nasazení u více zákazníků, a proto jednotlivé úpravy nemohou ovlivňovat chování základních principů. Cíl je tedy, za použití co nejmenšího úsilí a co nejmenších změn, požadované úpravy do systému implementovat. Tento celkový přístup k problematice rozšiřitelnosti systému by měl do budoucna přinášet hlavně finanční úspory a lepší udržovatelnost.

Kromě běžných problémů způsobených odlišností vývoje pro mobilní zařízení byl největší problém v odlišnosti jednotlivých zařízení. O této problematice pojednává následující kapitola.

5.1 Odlišnosti jednotlivých verzí OS

Pokud jde o operační systémy na mobilních zařízeních jedná se prakticky o dvě základní skupiny. Je to Windows CE a dále pak Windows Mobile. Oba tyto systémy mají i své rozdílné stupně výbavy a různé verze, ovšem v rámci těchto podkategorií žádné velké změny nejsou.

Windows CE je v podstatě modulární základ pro tvorbu dalších systémů či jejich verzí. Právě použitím některých komponent systému Windows CE vzniká nový systém Windows Mobile. Ten je používán převážně na zařízeních typu PDA a telefonech [5]. To je více méně základní jednoduchý fakt o Windows Mobile. Mělo by tedy platit, že to co obecně funguje na Windows CE, bude bez problémů fungovat i na Windows Mobile. Bohužel ovšem opak je pravdou, oba systémy mají svá specifika a dost často se jedná o diametrální rozdíly v chování.

5.1.1 Dialogy

Nejzásadnější rozdíl, který je vidět u Windows Mobile na první pohled, je celkové rozložení dialogů. V této verzi operačního systému Windows CE v podstatě neexistují klasické dialogy tak, jak je známe z plných verzí systému Windows na PC. Každé okno je zobrazeno přes celou obrazovku a nelze jej minimalizovat či zmenšovat. Navíc ani neobsahuje lištu s titulkem a tlačítky pro práci s dialogem a ani tlačítko pro zavření okna. To je integrováno do lišty start, která obyčejně bývá nahoře.

Zobrazení formulářů je tedy poněkud odlišné a je nutné různé pomocné dialogy, původně plánované v malém formuláři uprostřed obrazovky, zobrazovat jinak. Při vývoji to není nějaký zásadní problém, jen je nutno na tuto odlišnost brát zřetel z hlediska estetického.

Situace u Windows CE je v podstatě velice podobná klasickým Windows. Je zde start lišta, která může být skryta a veškeré dialogy mohou být normálně přesunovány, minimalizovány či jejich velikost libovolně měněna. Pokud zařízení disponuje dotykovou obrazovkou, práce s tímto systémem je opravdu skoro stejná jako například Windows XP [5].

5.1.2 SIP

Soft input panel [5] neboli SIP je technika používaná na dotykových zařízeních bez klávesnice pro vkládání znaků. Klasicky se jedná o softwarovou klávesnici zobrazenou na dotykové obrazovce. Poklepáním na znak je tento znak vložen do aktivního okna tak, jako by jej uživatel napsal na klávesnici. U vyvíjeného systému je při práci na PDA, které klávesnici většinou nemá, tato pomůcka více než jen užitečná, je dokonce nutná. Je potřeba například filtrovat výrobky dle názvu. To nám ovšem bohužel přináší jeden nepříjemný fakt. V názvech výrobků se vyskytují české znaky. Na zařízeních s anglickým systémem jako jsou například zařízení Motorola ovšem česká klávesnice není. Ta se vyskytuje pouze na českých PDA.

Na trhu lze narazit na několik firem nabízejících pro mobilní zařízení lokalizaci, včetně české klávesnice. Většina z nich je vyhovujících. Problém je, že nikde neexistuje česká klávesnice SIP pro zařízení s Windows CE, pouze Windows Mobile. Je pravda, že takových zařízení není mnoho, jedná se v podstatě pouze o terminály se čtečkami čárových kódů jako je Motorola či Partner, nicméně pro tato zařízení je aplikace primárně vyvíjena. Bylo tedy nutné implementovat vlastní SIP klávesnici s českými znaky, která bude fungovat na zařízeních s Windows CE. Soft input panely jsou programovány v nativním C++. Bylo tedy okopírováno chování klávesnice z českého PDA (tedy z Windows Mobile) a znovu implementováno pro Windows CE.

Jak bylo již dříve zmíněno, zařízení s dotykovým panelem musí umožňovat snadné ovládání celého systému tak, jako to umožňují zařízení s klávesnicí. Ta bohužel na většině PDA ale chybí, a proto celý princip virtuálních kláves zde není využitelný. Je nutné navrhnout nový způsob práce.

Místo implementace složitých menu byl navržen velice chytrý způsob ovládání, který využívá stávajících implementovaných technologií. Jedná se o vytvoření nového SIP panelu, který bude obsahovat nejběžnější příkazy jako je pohyb, ukládání dokladu, opuštění dokladu a podobně. V novém SIPu budou tedy zobrazena tato tlačítka a k nim se přiřadí různé klávesy, které nejsou běžným způsobem využitelné. Jedná se zejména o symboly a podobné znaky, které na standardní klávesnici ani napsat nelze. Při stisku každého z těchto tlačítek dojde k simulaci stisku dané klávesy. A nyní je právě okamžik, kdy stávající technologie může zafungovat. Vrstva virtuálních kláves totiž může být napojena na „stisk“ jakékoli klávesy. A to i stisk simulovaný a navíc stisk těchto symbolů

(tedy nesmyslných kláves). Tím pádem nedochází k ovlivnění standardních funkcí systému. Na tuto akci je pak možné již libovolně reagovat, dle možností frameworku.

Takovým způsobem bylo elegantně vyřešeno ovládání aplikace pomocí dotykového panelu. Naráz může být zobrazen pouze jeden SIP, nicméně to není problém. Každé zařízení s dotykovým panelem většinou obsahuje alespoň dvě či čtyři hardwarová tlačítka, na ně lze napárovat vyvolání těchto SIP klávesnic, tedy české klávesnice a ovládacího prvku pro framework.

5.2 Odlišnosti jednotlivých mobilních zařízení

Každé zařízení má svá určitá specifika, na která je potřeba brát ohled. Rozdíly se vyskytují zejména mezi různými výrobci zařízení, který také k ovládání a práci s těmito komponentami dodává vlastní knihovny, tak zvané SDK. Týká se to například Motoroly, která má SDK pro obsluhu čtečky čárových kódů [7], ale také klávesnice či indikačních LED diod. Veškerá závislost na konkrétních SDK je řešena mimo hlavní část systému a to ve speciálních modulech pro daného výrobce. Tím se stále udržuje nezávislost celého frameworku na konkrétním zařízení.

V této kapitole budou popsány hlavní problémy, kterým bylo v průběhu vývoje projektu věnováno více času.

5.2.1 Správa zařízení SD karty

Mobilní zařízení a obzvláště datové terminály jakými jsou výrobky Motoroly mají poměrně málo paměti. Navíc v systému Windows CE (na rozdíl od Windows Mobile) dojde při hard resetu ke smazání a znovu načtení velké části „pevného disku“ z ROM paměti. Ve skutečnosti je totiž hlavní adresářová struktura pouhým RAM diskem. Statická data lze ukládat do flash paměti, která je ale pomalá a hlavně co se kapacity týče opět více než omezená.

Proto je vhodné ukládat veškerá data (a to i vlastní program, který samozřejmě také nesmí být smazán) na SD, případně MMC kartu. Čtečku těchto karet dokonce ve formátu SDIO podporují skoro všechny zařízení. SDIO formát umožňuje do takového zařízení připojit přes SD kartu další komponentu jako je třeba bluetooth, wifi, ale i kamera či čtečka čárových kódů. To je nicméně v našem případě nedůležité. Problém s touto externí pamětí nastává při uspání zařízení. V takovém případě je SD karta také uspána a tím pádem odpojena. To s sebou přináší jednu nepříjemnou vlastnost. Při každém odpojení SD karty je tato odmontována z operačního systému a veškeré ukazatele na otevřené soubory včetně databáze jsou uzavřeny. Po opětovném probuzení zařízení celá aplikace spadne, protože se odvolává na neplatné ukazatele zdánlivě otevřených souborů. Tato vlastnost, tedy odpojování karty, je řešitelná na úrovni Windows, resp. ovladačů SD karty. Na zařízeních Motorola je naštěstí tohle podporováno a stačí v registrech systému nastavit příslušné údaje. Po tomto zásahu nedochází k odpojení SD karty po uspání zařízení. Bohužel nelze tento

princip aplikovat na MC1000, kde tedy není možné používat SD kartu a nebo je nutné zakázat uspávání zařízení.

5.2.2 MC1000 a virtuální klávesy

Na zařízení MC1000 ze záhadného důvodu nefungovaly virtuální klávesy, resp. nedařilo se napojit na vrstvu operačního systému, která zachytává stisky kláves, tzv. hook funkcemi. Po nemalém pátrání bylo zjištěno, že datové terminály Motorola mají vlastní aplikace starající se o klávesnici. Takovou aplikaci bylo nutné zlikvidovat a virtuální klávesy začaly fungovat bez problémů.

Zvláštností všech terminálů Motorola je také to, že veškeré klávesy mohou být mapovány speciálním software od výrobce. A to včetně stisků kláves ve všech různých módech klávesnice. Je například možné přemapovat funkci klávesy „*“ na klávesu „-“. To je samozřejmě výhodné, protože terminál může být plně odladěn pro práci s vyvíjenou aplikací.

5.3 Instalace systému

Instalace systému se dělí na dvě části a to opět na komunikační infrastrukturu a na vlastní terminál. Pro komunikační infrastrukturu není problém udělat instalační soubory, tak jako pro každou jinou aplikaci běžící pod OS Windows. U mobilního zařízení je to ale horší. Vytvořit instalační soubory možné je, nicméně jak již bylo zmíněno výše, po cold/hard resetu dojde k vymazání celého zařízení. Proto musí být tyto instalační soubory umístěni ve statické paměti a navíc je nutné po smazání celé zařízení znovu nainstalovat. To se netýká jen vlastní aplikace, ale také všech ovladačů a většinou i .NET Frameworku, který nebývá součástí standardní distribuce.

Pro tento případ bylo vyvinuta speciální aplikace, která se pouští po každém startu systému. V případě, že se jedná o první spuštění, nainstaluje všechny potřebné součásti a poznačí si, že již byla spuštěna. V rámci toho probíhá instalace .NET frameworku, ovladačů zařízení, SIP klávesnice a ovládacího prvku systému, ale například také českých regionálních nastavení a různých dalších konfigurací.

6 Závěr

Cílem diplomové práce byla implementace programových prostředků systému pro sběr dat na datovém terminálu Motorola MC1000 a podobných zařízeních. Při implementaci bylo použito výsledků předcházejícího semestrálního projektu, který se zabýval analýzou a návrhem řešení celého systému. Tento návrh byl v několika bodech doplněn či pozměněn. Byla implementována architektura systému obecně propojitelného na jakýkoli další software zpracovávající výstupní data. Pro další zpracování musí být vždy implementován převodní můstek do konkrétního hostitelského systému, tento můstek byl v rámci práce vyvinut pro software Pohoda společnosti Stormware s.r.o.

Pro realizaci systému byly vybrány poměrně zajímavé a dnes velmi rozšířené a podporované technologie jako jsou XML dokumenty a jejich zpracování pomocí XSLT. Data jsou ukládána do SQL serverů, čímž je usnadněna manipulace s nimi. Na datovém terminálu běží operační systém Windows CE či Windows Mobile, který podporuje .NET Compact Framework, a proto jsou zde k dispozici prakticky všechny standardní komponenty potřebné pro vývoj běžné aplikace. Pro obsluhu nestandardních částí terminálu, jakým je například scanner čárových kódů, jsou dodávány komponenty ve formě SDK balíčku od výrobce zařízení.

Byl implementován robustní systém sestávající z několika součástí. Celá architektura je velice dobře nastavitelná a modifikovatelná dle potřeb konkrétního zákazníka. Konfigurace se vždy provádí pomocí XML souborů validních dle daných schémat. Jakékoli nastavení systému je poměrně složité, protože v hodně případech se jedná o téměř vývojářskou práci. Například nastavení xpath filtrů vyžaduje znalosti XML souborů, práce se jmennými prostory i detailní znalost struktury jednotlivých elementů. Tento požadavek je ovšem vyvážen tím, že zákaznické úpravy je často možno řešit pouze těmito konfiguracemi. Sytém je tedy navržen tak, že jeho instalaci bude vždy provádět zaškolený technik. Uživatel sám do konfigurace zasahovat nebude.

Data typu číselníků jsou z hostitelského systému přes komunikační infrastrukturu s možností různých modifikací nahrána do mobilního zařízení. S těmito daty je pracováno jako s XML balíčky a jsou zde použity pro zpracování technologie xpath filtrů. Na mobilním zařízení jsou data uložena do SQL databáze a uživatel s nimi pracuje. Vytvořené doklady jsou opět ve formě XML balíčku synchronizovány zpět komunikační infrastrukturou do hostitelského systému.

Systém byl otestován v laboratorním prostředí a většina drobných a všechny velké závady byly odstraněny. Pomocí mobilního zařízení je možné vytvářet do hostitelského systému, kterým je zde Pohoda společnosti Stormware s.r.o., nejrůznější účetní doklady jako je faktura, výdejka, příjemka, převodka, pokladní doklad, ale například i inventura. Při vývoji se ukázala jako zásadně nedostatečná XML brána Pohody a na jejích úpravách a odstranění nedostatků bylo spolupracováno s firmou Stormware.

Po laboratorních testech byl systém nasazen u dvou zákazníků, kteří jej v současné době již více než měsíc používají. Postupně dochází k odstraňování některých nových drobných problémů, nicméně základ systému je plně funkční. Zákazníci používají zařízení pro vytváření pravidelných inventur skladů či jako zařízení umožňující mobilní prodej obchodním reprezentantům firmy, kteří také přímo z nich tisknou doklady pro zákazníky.

Společně s vývojem XML brány Pohody lze vyvíjet i další funkce celého systému. Například propojování dokladů jako jsou objednávky a faktury či pokladní doklady. Tyto nové vlastnosti by systém učinily pro zákazníky ještě zajímavější, a proto jejich implementace bude dalším krokem ve vývoji.

Další nezávislou řadou je vývoj napojení na jiné hostitelské systémy. Původní myšlenka použití celého systému na mobilním zařízení zřejmě zachována nebude. Některé moduly pro práci s daty jsou totiž natolik specifické a přizpůsobené systému Pohoda, že jejich použití je takřka vyloučeno. Situace je tedy taková, že bude využita celá komunikační infrastruktura společně s frameworkem na mobilním zařízení. Moduly pracující s daty budou muset být implementovány nezávisle pro další systém. To je ale poměrně malá změna, protože tyto moduly obsahují pouze několik málo formulářů a ne mnoho metod pro vlastní práci.

Systém je tedy plně funkční a má velice dobré vyhlídky na další vývoj a dokonce i na použití ve stávajícím stavu.

Literatura

[1] Arlow Jim: *UML 2 a unifikovaný proces vývoje*. Brno, Computer Press 2007.

[2] Evjen B., Glynn J., Jones A., Nagel Ch., Skinner M., Watson K.: *C# 2005 Programujeme profesionálně*, Brno, Computer Press 2006.

[3] Symbol Technologies, Inc.: *MC1000 Integrator Guide*. New York, Symbol Technologies, Inc 2005.

[4] Stormware s.r.o.: *Ekonomický systém Pohoda – Příručka uživatele*. Jihlava, Stormware s.r.o. 2006.

[5] Bollng Douglas: *Programming Microsoft Windows CE .NET, Third Edition*. Washington, Microsoft Press 2003.

[6] A. Benadiková, Š. Mada, S. Weinlich: *Čárové kódy - automatická identifikace*. Praha, Grada 1994.

[7] Motorola, Inc.: *Symbol Mobility Developer Kit for .NET Help v1.7*. New York, Motorola, Inc. 2008.

[8] W3C Consortium: XML *Path Language (XPath)*. http://www.w3.org/TR/xpath, cit. květen 2009.

[9] Stormware s.r.o.: *XML Komunikace.* http://stormware.cz/xml, cit. leden 2009.

[10] Jiří Kosek: *XSLT v příkladech*. http://www.kosek.cz/xml/xslt, cit. únor 2009.

[11] Microsoft Corporation: *SQL Server Compact 3.5.* http://www.microsoft.com/Sqlserver/2005/en/us/compact.aspx, cit. květen 2009.

[12] FieldSoftware Products: *PrinterCE.NetCF Developer's Guide.* http://www.fieldsoftware.com/PrinterCE_NetCF_Docs.htm, cit. leden 2009.

[13] Microsoft Corporation: *Microsoft Synchronization Services for ADO.NET Books Online*. Washington, Microsoft Corporation 2007.

Seznam příloh

Příloha 1: CD se zdrojovými texty a instalačními soubory.